64'er

TIPS & TRICKS SONDERHEFT

SONDERHEFT 2/86 ÖS 100,-/Sfr. 14,- Lit. 12000/hfl. 18,-/dkr. 68,- DM 14,-

Markt&Technik

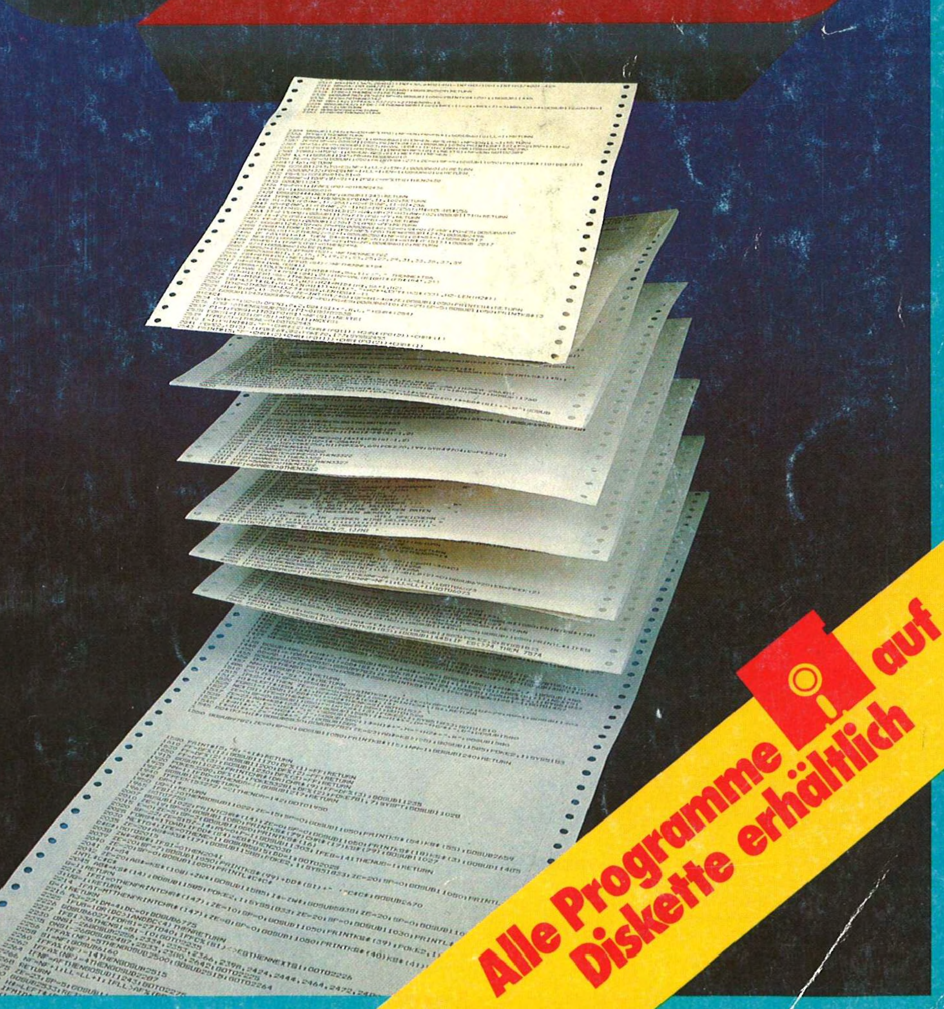
64'er

TIPS & TRICKS

Super Listings

Ausführliche Grundlagen

- ★ Floppy
- ★ Datasette
- ★ Drucker
- ★ Peeks & Pokes
- ★ Einzeiler
- ★ Sprites
- ★ Grafik
- ★ Musik
- ★ Basic
- ★ Dateiverwaltung
- ★ Fehlersuche
- ★ Assembler



Alle Programme auf Diskette erhältlich





Das Salz in der Suppe

Tips & Tricks – für viele ein Begriff mit fast magischer Anziehungskraft! Ihr Kennzeichen: kurz und interessant. Der Commodore 64 ist nicht ohne Grund der meistverkaufte Computer. War es am Anfang das sagenhafte Preis-/Leistungs-Verhältnis, kam mit der Zeit eine immer noch steigende Anzahl von Programmen und Geräten hinzu (und last not least viele Bücher und Zeitschriften wie das 64'er-Magazin). Andererseits verführt die Mischung aus umfangreicher Grundausstattung (Speicherplatz, Grafik und Sound) und spartanischem Betriebssystem (Basic) bei großer Zugänglichkeit (User-Port, Expansion-Port, Grafik- und Soundchips etc.) zu umfangreichen und bei manchen zu enthusiastischen Aktivitäten. Endlich gab es ein Gerät, mit dem man nicht nur spielen und »ernsthaft« arbeiten konnte (Textverarbeitung, Dateiverwaltung etc.), sondern bei dem die Kreativität gefördert wurde, und zwar sowohl für Anfänger als auch für Profis. Was man brauchte und (ohne viel Geld) nicht bekommen konnte, wurde eben selbst gemacht. Bei dieser Beschäftigung mußte und muß man sich mit einer Vielzahl von Problemen befassen, die man, wenn man sie gelöst hat, anderen C 64-Besitzern natürlich nicht vorenthalten möchte. Die Anzahl und die Art der immer wieder neuen Programme und Programmhilfen, der Tools und Utilities führen immer wieder zu Überraschungen und verführen zu neuen Ideen. In diesem 64'er Tips & Tricks-Sonderheft findet deshalb jeder etwas interessantes. Lassen Sie sich ebenfalls verführen!

Am laufenden Band

Es ist kaum möglich, alle Tips & Tricks, die bisher veröffentlicht wurden, auswendig zu kennen. Oft weiß man auch gar nicht, daß es zu einem bestimmten Problem eine ganz einfache Lösung gibt. Manchmal kommt einem der Verdacht oder die vage Erinnerung, daß man schon irgendwo etwas darüber gelesen hat, nur – wo? Schließlich gibt es nicht nur das erste 64'er Tips & Tricks-Sonderheft (erschienen 1984), sondern auch eine stetig steigende Zahl 64'er-Ausgaben, in denen auch jeden Monat neue Tips & Tricks erscheinen.

Bewährtes...

Es ist also gar nicht so abwegig, ein Sonderheft zu machen, in dem neben vielen neuen Beiträgen auch die besten bisher veröffentlichten Tips & Tricks zusammengefaßt werden (»Die besten Tips & Tricks aus 1984/85« und »Einzeiler«). Dabei haben wir uns nicht nur auf die kurzen Leckerbissen beschränkt, sondern auch Grundlagenwissen nicht vergessen. Wer die ersten 64'er-Ausgaben von 1984 nicht besitzt, wird zum Beispiel kaum etwas ahnen von den sagenhaften Eigenschaften der »synthetischen Steuerzeichen«. Auch das Wissen um die Tastencodes, die Tastaturabfrage und die sich daraus ergebenden Möglichkeiten für Programmierer fand 1984 großen Anklang.

...und Neues

Natürlich gehört in ein Tips & Tricks-Sonderheft auch »fertige Kost«, vor allem aus dem Bereich Tools und Utilities. Gemeint sind Programme, die die Arbeit mit dem Computer erleichtern. Sie finden deshalb ausgezeichnete Programme aus den Themenbereichen Musik, Grafik, Floppy und Programmierung. Für jeden ist etwas dabei. Wenn Ihnen dieses Heft gefallen hat, freuen wir uns natürlich über Ihre Zuschriften. Aber auch, wenn Sie Verbesserungsvorschläge haben oder wenn Sie etwas vermissen – halten Sie sich mit Ihrer Meinung nicht hinter dem Berg! Denn sie ist für uns das Salz in der Suppe.

(Georg Klinge)

Diskettenservice

Wer keine Zeit oder Lust hat, alle Programme selbst in mühevoller Kleinarbeit abzuschreiben, kann wieder auf den bewährten Diskettenservice zugreifen. Alle Programme, die mit dem Diskettensymbol im Inhaltsverzeichnis gekennzeichnet sind, gibt's auf Diskette.

Bestell-Nr. L6 86 S2D **29,90 Mark**
inkl. MwSt.



Bestellungen aus der Schweiz bitte direkt an: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042/41 56 56. Bestellungen aus Österreich bitte direkt an: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Str. 24, 1091 Wien, Tel. 0222/48 15 38-0.

PROGRAMM-SERVICE

Bestellungen aus anderen Ländern bitte per Auslandspostanweisung! Achtung: Nicht die eingeheftete Zahlkarte verwenden!

Programme aus früheren Ausgaben:

Sonderheft 1/86: C 128er
Diskette
Bestell-Nr. L6 86 S2D **DM 29,90***

Sonderheft 8/85: Assembler
Diskette
Bestell-Nr. L6 85 S8D **DM 29,90***
Kassette
Bestell-Nr. L6 85 S8K **DM 19,90***

Sonderheft 7/85: Professionelle Anwendungen
2 Disketten
Bestell-Nr. L6 85 S7D **DM 34,90***
4 Kassetten
Bestell-Nr. L6 85 S7K **DM 34,90***

Sonderheft 6/85: Top-Themen
2 Disketten
Bestell-Nr. L6 85 S6 **DM 34,90***

Sonderheft 5/85: Floppy, Datasette
Diskette
Bestell-Nr. L6 85 S5D **DM 29,90***
Kassette
Bestell-Nr. L6 85 S5K **DM 19,90***

Sonderheft 4/85: Grafik
Bestell-Nr. L6 85 S4A **DM 29,90***

Sonderheft 3/85: Spiele
Beide Disketten in einem Paket!
Verwenden Sie **nur** diese Bestell-Nr.:
Bestell-Nr. L6 85 S3A **DM 34,90***

Sonderheft 2/85: Abenteuerspiele
Bestell-Nr. L6 85 S2 **DM 34,90***

Sonderheft 1/85: Tips & Tricks (2. überarb. Auflage)
Floppy-Utilities
Bestell-Nr. CB 023 **DM 29,90***
Hilfsprogramme
Bestell-Nr. CB 024 **DM 29,90***

Ausgabe 2/86
Bestell-Nr. L6 86 02D **DM 29,90***

Ausgabe 1/86
Bestell-Nr. L6 86 01D **DM 29,90***

Ausgabe 12/85
Diskette
Bestell-Nr. L6 85 12D **DM 29,90***
Kassette
Bestell-Nr. L6 85 12K **DM 29,90***

Programm	Seite
Checksummer V3	S. 54
MSE V1 .0	S. 54
Old für C 128	S. 43
Chemie-Assistent	S. 57
SMU	S. 68
Hyperscreen	S. 76
Grafik-80	S. 80
Seeschlacht	S. 93
Eprom-Automat	S. 93
Tipp-Utility	S. 99
Floppymonitor	S. 105
Auto.OBJ	S. 108
Bildsch.Langsam	S. 107
Taschenrechner	S. 107
Code-ASCII	S. 107
88-Zeichen	S. 107
Frogger	S. 106
Scroll n. unten	S. 106
Zahlenraten	S. 108
Auto-Befehl	S. 107
SWAP	S. 153
BSP.-Quelltext	S. 169

Ausgabe 11/85
Bestell-Nr. L6 85 11A **DM 29,90***
Commodore 64

Programm	Seite
Checksummer V3	S. 54
MSE	S. 54
Koala-Painter Hardcopy	S. 39
Lyrik-Maschine (AdM)	S. 55
Hypra-Platos (LdM)	S. 61
Profiprint	S. 71
Apfelmännchen	S. 80
Block Out	S. 84
Spritekill	S. 86
Screen-Dump	S. 88
Pseudo-IRQ	S. 88
INPUT-Routine	S. 90
Synthetische Melodien	S. 95
Hypra-Ass Ergänzung	S. 96
Reassembler	S. 97
Vier Betriebssysteme	S. 105
Spiralen	S. 151
HiRes-Spiralen	S. 151
Plotter-Spiralen	S. 151
Fensterrose	S. 151
HiRes-Fensterrose	S. 152
Plotter-Fensterrose	S. 152
Abweichungen	S. 152
Funktionenplot	S. 153
3D-Programm	S. 154
REM-Text-Killer	S. 158
Sound-Editor mit Sequencer	S. 158
Sequencer-Ergänzung	S. 159
Testsong	S. 159
Sequenzgenerator	S. 159

Ausgabe 10/85
Leider hat sich in die Bestell-Nummer der letzten Programm-Service-Anzeige ein Druckfehler eingeschlichen. Die korrigierte Bestell-Nummer lautet:
L6 85 10A **DM 29,90***
Commodore 64

Programm	Seite
Check V3 Dez 64	S. 54
MSE V1.0	
Floppy-Adjust	S. 32
Eprom-Trans	S. 42
Schreiberling	S. 54
Cursus Latinus (AdM)	S. 57
Hypra-Text (LdM)	S. 67
Pacman	S. 76
Programm GEN	S. 86
SMON+	S. 87
Sequenzer	S. 129
Musik	S. 129
Alarmanlage	S. 132
Codeschloß	S. 132
Crossreference verb. Version	S. 83

Ausgabe 9/85
Bestell-Nr. L6 85 09A **DM 29,90***
Commodore 64

Programm	Seite
Sound-Machine	S. 23
Noteneingabe	S. 24–25
Sound Master	S. 32
Ringmod	S. 32
Moonlight	S. 33
SYNC	S. 33
Prüfungsfragen (AdM)	S. 55–58
Schlüssel (LdM)	S. 59–61
Disk Designer	S. 70–72
Blinker	S. 73
Logelei-1/2	S. 118
Lichtgr.	S. 122
Mischsort	S. 127
Block Busters	S. 159
X-Gleichung	S. 159
Musik-Tool	S. 159

Ausgabe 8/85
Bestell-Nr. L6 85 08A **DM 29,90***
Commodore 64

Programm	Seite
Quicksort	S. 142
Procedure	S. 78
Hypra-Save	S. 79
Uhr	S. 22
NEWEA2 (AdM)	S. 60
Disk-Monitor	S. 84
Maskengenerator	S. 87
Bit-Map	S. 81
HiRes3-Komplett	S. 159
Forth-Compiler (LdM)	S. 63
Vocabulary	S. 69
Schach	S. 74
Extern-Kurs	S. 147
Sprites	S. 44
Hypra-Zusatz	S. 25
Hi-Text 2.0	S. 71

Ausgabe 7/85
Bestell-Nr. L6 85 07A **DM 29,90***

Ausgabe 6/85
Bestell-Nr. L6 85 06A **DM 29,90***

Ausgabe 5/85
Bestell-Nr. L6 85 05A **DM 29,90***

Ausgabe 4/85
Bestell-Nr. L6 85 04A **DM 29,90***

Ausgabe 3/85
Bestell-Nr. L6 85 03A **DM 29,90***

Ausgabe 2/85
Bestell-Nr. L6 85 02A **DM 29,90***

Ausgabe 1/85
Bestell-Nr. L6 85 01A **DM 29,90***

Ausgabe 12/84
Bestell-Nr. CB 022 **DM 29,90***

Ausgabe 11/84
Bestell-Nr. CB 020 **DM 29,90***

Ausgabe 10/84
Bestell-Nr. CB 019 **DM 29,90***

Bedeutung der Abkürzungen

*LdM = Listing des Monats
*AdM = Anwendung des Monats
*SB = Simons Basic
*GV = Grundversion
*GV > = alle Speicherversionen können verwendet werden (einschließlich GV)
*3K = 3-KByte-Speichererweiterung wird benötigt
*8K > = Speichererweiterung größer als 8 KByte wird benötigt
*UPB = Unterprogrammbibliothek

* Alle Preise inklusive Mehrwertsteuer.

Bitte verwenden Sie für Ihre Bestellung **nur** die eingeheftete Postscheck-Zahlkarte zur Überweisung des Rechnungsbetrags.

Fehlende Hefte erhalten Sie bei: Markt & Technik Vertrieb 64'er Hans-Pinsel-Str. 2 8013 Haar

64er-online.de
64er-online.net

INHALT

Vorwort

Musik

Grundlagen

Grafik

Tips & Tricks-Listings

Spiel

64er-online.de
64er-online.net

Checksummer 64 V3

Der Checksummer 64 V3 überprüft jede Basic-Zeile direkt nach der Eingabe, erkennt Fehleingaben sowie Vertauschungen von Ziffern und erspart eine aufwendige Fehlersuche.

Der Checksummer 64 V3 ist ein kleines Maschinenprogramm, das Sie sofort unterrichtet, ob Sie die jeweilige Programmzeile korrekt eingegeben haben.

So gehen Sie vor:

1. Programm abtippen und speichern.

2. Starten mit RUN

3. Nach kurzer Zeit sehen Sie am Bildschirm: CHECKSUMMER 64, CHECKSUMMER AKTIVIERT, AUSSCHALTEN MIT POKE 1,55, ANSCHALTEN MIT POKE 1,53, READY.

4. Anschalten des Checksummer 64 V3 mit POKE 1,53.

5. Test: Geben Sie in einer freien Zeile ein: »1 REM« und drücken die RETURN-Taste. Am Bildschirm oben links sollten Sie die Prüfsumme <63> sehen.

6. Geben Sie ein Listing aus unserem Heft ein. Nach jeder Zeile wird die Zahl, die im Listing in Klammern < > steht, in den Bildschirm eingeblendet. Stimmen die Zahlen nicht überein, so liegt vermutlich ein Eingabefehler vor. **Die Zahl in den Klammern, und auch die Klammern selbst, dürfen beim Abtippen nicht mit eingegeben werden!**

7. Der Checksummer 64 V3 bemerkt auch Vertauschungen von Zahlen und Buchstaben, aber nicht das Fehlen (oder Hinzufügen) von Leerzeichen.

8. Unsere Basic-Listings enthalten keine Steuerzeichen mehr. Diese werden ersetzt durch Klartext und stehen zwischen geschweiften Klammern. Deshalb sind weder die Klammern noch was dazwischen steht, abzutippen, sondern die in Tabelle 1 aufgeführten Tasten zu drücken. Auf Ihrem Bildschirm erhalten Sie dann wieder die entsprechenden Grafikzeichen.

9. Alle Grafikzeichen werden ebenfalls ersetzt durch unterstrichene oder überstrichene Großbuchstaben.

Unterstrichene Buchstaben bedeuten, daß Sie die SHIFT-Taste und den angegebenen Buchstaben drücken müssen, überstrichene jedoch die Commodore-Taste mit dem Buchstaben. Auch hier erhalten Sie am Bildschirm das entsprechende Grafikzeichen und nicht etwa das im Listing erkennbare Zeichen.

Die Leerzeichen zwischen den einzelnen Basic-Befehlen können beim Abtippen entfallen (ohne Einfluß auf die Checksumme zu nehmen). Dies ist besonders bei speicherkritischen Programmen wichtig. Ebenso müssen Zeilen, die mehr als 80 Zeichen pro Zeile enthalten, mit den bekannten Abkürzungen für die Basic-Befehle (siehe auch das Handbuch zum C 64, Anhang D, Seite 130) eingegeben werden.

Sie können die Programme auch weiterhin ohne den Checksummer eintippen.

(F. Lonczewski/gk)

Hinweis: {13 SPACE} bedeutet 13mal die Leertaste drücken

```
9 REM ********************************
10 PRINT"{CLR,11SPACE,RVSON}CHECKSUMMER 64
   V3{RVOFF}"
11 PRINT"{2DOWN,9SPACE}EINEN MOMENT, BITTE
   ..."
12 FOR I=828 TO 864:READ A:POKE I,A:PS=PS+
   A+1:NEXT I
13 IF PS<>5802 THEN PRINT"PRUEFSUMMENFEHLE
   R IN ZEILEN 20-22":END
14 SYS 828:PS=0:FOR I=58464 TO 58583:READ
   A:POKE I,A:PS=PS+A+1:NEXT I
15 IF PS<>16267 THEN PRINT"PRUEFSUMMENFEHL
   ER IN ZEILEN 22-30":END
16 POKE 1,53:POKE 42289,96:POKE 42290,228
17 PRINT"{4DOWN,9SPACE}CHECKSUMMER AKTIVIE
   RT."
18 PRINT"{2DOWN}AUSSCHALTEN : POKE1,55"
19 PRINT"{DOWN}ANSCHALTEN{2SPACE}: POKE1,5
   3":NEW
20 DATA 169,0,133,254,162,1,189,93,3,133,2
   55,160,0,177,254
21 DATA 145,254,136,208,249,230,255,165,25
   5,221,95,3,208,238,202
22 DATA 16,230,96,160,224,192,0,160,2,169,
   0,170,133,254,177
23 DATA 95,240,40,201,32,208,3,200,208,245
   ,133,255,138,41,7
24 DATA 170,240,14,72,165,255,24,42,105,0,
   202,208,249,133,255
25 DATA 104,170,232,165,255,24,101,254,133
   ,254,76,111,228,192,4
26 DATA 48,219,198,214,165,214,72,162,3,16
   9,32,157,1,4,189
27 DATA 212,228,32,210,255,208,12,0,92,72,
   32,201,255,170,104
28 DATA 144,1,138,96,202,16,228,166,254,16
   9,0,32,205,189,169
29 DATA 62,32,210,255,104,133,214,32,108,2
   29,169,141,32,210,255
30 DATA 76,128,164,9,60,18,19
```

© 64'er

Der Checksummer 64 V3 erkennt auch Vertauschungen von Zahlen

CTRL steht für Control-Taste, so bedeutet [CTRL-A], daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:

Steuerbefehl	Taste
{DOWN}	Taste neben rechtem Shift, Cursor unten
{UP}	Shift-Taste & Taste neben rechtem Shift; Cursor hoch
{CLR}	Shift-Taste & 2. Taste ganz rechts oben
{INST}	Shift-Taste & Taste ganz rechts oben
{HOME}	2. Taste von ganz rechts oben
{DEL}	Taste ganz rechts oben
{RIGHT}	Taste ganz rechts unten
{LEFT}	Shift-Taste & Taste unten rechts
{SPACE}	Leertaste
{F1} bis {F8}	Funktionstasten
{RETURN}	Shift-Taste & Return
{BLACK}	Control-Taste & 1
{WHITE}	Control-Taste & 2
{RED}	Control-Taste & 3
{CYAN}	Control-Taste & 4
{PURPLE}	Control-Taste & 5
{GREEN}	Control-Taste & 6
{BLUE}	Control-Taste & 7
{YELLOW}	Control-Taste & 8
{RVSON}	Control-Taste & 9
{RVOFF}	Control-Taste & 0
{ORANGE}	Commodore-Taste & 1
{BROWN}	Commodore-Taste & 2
{LIG.RED}	Commodore-Taste & 3
{GREY 1}	Commodore-Taste & 4
{GREY 2}	Commodore-Taste & 5
{LIG.GREEN}	Commodore-Taste & 6
{LIG.BLUE}	Commodore-Taste & 7
{GREY 3}	Commodore-Taste & 8

Tabelle 1. Die Steuerbefehle in den Listings

64er-online.de

MSE – Abtippen sicher und leicht gemacht

Ähnlich wie der »Checksummer« ist auch der MSE ein Hilfsmittel bei der Eingabe von Listings, diesmal jedoch bei reinen Maschinensprache-Programmen.

Im Gegensatz zum »Checksummer« aber ist die Eingabe nicht ohne den MSE möglich. Der MSE verringert die Tipparbeit um ein Drittel und schließt Fehleingaben vollkommen aus. Außerdem können Sie die Werte blind eingeben, ohne andauernd auf den Bildschirm schauen zu müssen. Dies wird durch akustische Meldungen realisiert.

MSE ist ein Maschinenspracheditor, mit dem ein Vertippen ausgeschlossen ist. Eine abgetippte Zeile wird nur angenommen, wenn sie richtig ist. Eine Checksumme am Ende jeder Zeile prüft, ob die richtigen Werte in der richtigen Zeile an der richtigen Stelle stehen. Wenn nicht, ertönt ein Warnsignal, und man beseitigt den Fehler.

War die Zeile korrekt, erklingt ein Gong, und die nächste Zeilennummer wird ausgegeben. Damit ist also auch »blindes« Eintippen möglich; Sie können sich voll auf den Text konzentrieren.

So arbeitet man mit MSE

Laden und starten Sie MSE. Zuerst wird der Programmname und die Start- und Endadresse erfragt. **Diese Angaben entnehmen Sie dem Kopf des jeweiligen abgedruckten Listings.** MSE meldet sich dann mit der Zeilennummer der ersten Zeile. Wenn Sie die Zeile richtig eingegeben haben, erscheint die nächste Zeilennummer und so weiter bis zum Ende. Zum Schluß wird das fertige Programm mit »CTRL-S« auf Diskette oder Kassette abgespeichert. Dazu sind keine weiteren Angaben mehr erforderlich. Das Programm kann dann ganz normal wieder geladen und gestartet werden.

Wenn Sie nicht alles auf einmal tippen wollen, können Sie jederzeit unterbrechen und den eingetippten Teil mit »CTRL-S« abspeichern. Wollen Sie weiterarbeiten, laden und starten Sie MSE wieder.

Geben Sie auf die Frage nach der Startadresse aber jetzt »L« ein, um Ihr Teilprogramm zu laden. Jetzt können Sie mit »CTRL-N« die Adresse eingeben, an der Sie weitertippen müssen. Wenn Sie sich nicht gemerkt haben, wie weit Sie gekommen sind, geben Sie nach dem Laden »CTRL-M« ein.

Auf die Frage nach der Startadresse antworten Sie mit der Anfangsadresse, die links in der Kopfzeile auf dem Bildschirm steht. Nun wird Ihr Programm aufgelistet. Mit »SPACE« wird das Listen fortgesetzt, mit »STOP« abgebrochen. Das Ende Ihres Programmteils erkennen Sie sehr einfach daran, daß nur noch der Wert »AA« in der Zeile steht. Die Adresse dieser Zeile müssen Sie anschließend mit »CTRL-N« eingeben. Das Programm ist nur mit »STOP/RESTORE« zu verlassen. Speichern Sie aber vorher unbedingt immer Ihren Text ab.

Hinweise zum Abtippen

Vor dem Abtippen oder späteren Wiederladen des MSE-**Laders** müssen Sie unbedingt folgende Zeile eingeben:

POKE 43,1: POKE 44,32: POKE 8192,0: NEW

Den MSE-**Lader** brauchen Sie nur einmal. Nach erfolgreichem Abtippen und Starten mit RUN geht der Lader verloren und es wird das endgültige Programm MSE V1.0 erzeugt. So gehen Sie vor:

Starten Sie das Programm mit RUN. Fehlerhafte Zeilen werden angezeigt und müssen korrigiert werden, bis der Lader zum »READY« durchläuft. Jetzt müssen Sie das fertige MSE-Programm speichern. Dazu brauchen Sie nur »RETURN« zu drücken, weil die erforderlichen Angaben schon auf dem Bildschirm stehen. (Kassettenbesitzer müssen in Zeile 343 die letzte Zahl in »1« abändern.) Ab jetzt können Sie »MSE V1.0« direkt, also ohne den DATA-Lader, benutzen. MSE V1.0 wird ganz normal mit »,8« geladen (keine POKEs notwendig).

(N. Mann/D. Weineck/gk)

MSE-Befehle:

DEL	löscht die letzte Eingabe.
CTRL-S	speichert das eingetippte Programm ab.
L oder CTRL-L	lädt ein Programm. Start- und Endadresse werden automatisch ermittelt.
CTRL-M	listet den Speicherinhalt. Abbruch mit STOP-Taste, weiter mit Leertaste.
CTRL-N	erlaubt die Eingabe einer neuen Adresse zum Weitertippen.
CTRL-P	gibt ein MSE-Listing auf dem Drucker aus.

```
100 REM ****************************      <091>
110 REM *                          *      <159>
120 REM *       M S E   LADER      *      <206>
130 REM *                          *      <179>
220 REM ****************************      <211>
230 REM                                   <036>
240 DIM H(75): FOR I=0 TO 9               <113>
250 H(48+I)=I: H(65+I)=I+10:NEXT          <041>
260 FOR I=2048 TO 3755 : READ A$          <198>
270 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1)) <199>
280 D=H(H)*16+H(L):S=S+D:POKE I,D         <219>
290 A=A+1:IF A<20 THEN NEXT:A=-1          <141>
300 PRINT " ZEILE:";1000+Z;               <011>
310 READ V :Z=Z+1:IF V=S THEN 330         <218>
320 PRINT"PRUEFSUMMENFEHLER !":STOP       <138>
330 IF A<0 THEN 341                       <221>
340 S=0:A=0:PRINT:NEXT                    <046>
341 PRINT"{CLR}P0̲43,1:P0̲44,8:P0̲45,172:P0̲46
    ,14                                   <010>
342 POKE 631,19:POKE 632,13:POKE 633,13:PO
    KE 198,3                              <249>
343 PRINT"{3DOWN}SAVE"CHR$(34)"MSE V1.0"CH
    R$(34)",8                             <171>
344 END                                   <092>
1000 DATA 00,0B,08,0A,00,9E,32,30,36,31,00
     ,00,00,A2,08,A9,36,85,A4,A9, 1247    <119>
1001 DATA 08,85,A5,A9,00,85,A6,A9,B0,85,A7
     ,A0,00,B1,A4,91,A6,C8,D0,F9, 2888    <054>
1002 DATA E6,A5,E6,A7,CA,D0,F2,A9,36,85,01
     ,4C,00,B0,20,D1,B1,A9,06,8D, 2787    <144>
1003 DATA 21,D0,A9,03,8D,20,D0,8D,86,02,A0
     ,B3,A9,74,20,FF,B1,A0,B3,A9, 2667    <237>
1004 DATA B9,20,FF,B1,A0,00,20,CF,FF,99,01
     ,02,C8,C9,0D,D0,F5,88,F0,D2, 2912    <217>
1005 DATA C0,0F,90,02,A0,0E,8C,00,02,20,EA
     ,B1,A0,B3,A9,CF,20,FF,B1,20, 2323    <013>
1006 DATA 8E,B4,85,FC,85,62,20,8E,B4,85,FB
     ,85,61,20,A7,B4,D0,20,A0,B3, 2864    <199>
1007 DATA A9,E5,20,FF,B1,20,8E,B4,85,60,20
     ,8E,B4,85,5F,20,A7,B4,D0,0A, 2624    <091>
```

Der MSE zum bequemen Abtippen von Maschinenprogrammen

64er-online.de 64er-online.net

```
1008 DATA A5,61,C5,5F,A5,62,E5,60,90,06,20
     ,43,B3,4C,3A,B0,A9,AA,A0,00, 2379   <167>
1009 DATA 91,FB,E6,FB,D0,02,E6,FC,20,3F,B2
     ,90,EF,4C,FB,B4,A2,02,86,58, 3118   <152>
1010 DATA A9,A6,A0,9D,20,F2,B1,20,E4,FF,F0
     ,FB,C9,30,90,0C,C9,47,B0,08, 2970   <231>
1011 DATA C9,3A,90,0B,C9,41,B0,07,C9,14,D0
     ,0F,4C,0B,B1,20,D2,FF,A6,58, 2322   <121>
1012 DATA 95,F7,C6,58,D0,D2,60,AE,8D,02,F0
     ,26,C9,0C,D0,03,4C,0B,B6,C9, 2685   <057>
1013 DATA 13,D0,03,4C,8B,B5,C9,0D,D0,03,4C
     ,BA,B4,C9,10,D0,03,4C,68,B5, 2282   <225>
1014 DATA C9,0E,D0,06,20,5F,B4,4C,64,B1,4C
     ,92,B0,A5,F9,20,02,B1,0A,0A, 2132   <208>
1015 DATA 0A,0A,85,F9,A5,F8,20,02,B1,05,F9
     ,60,C9,3A,90,02,69,08,29,0F, 1950   <092>
1016 DATA 60,A6,59,E0,08,90,1F,A6,58,E0,02
     ,B0,06,20,D2,FF,4C,8E,B0,C6, 2509   <188>
1017 DATA 59,A0,14,A9,92,20,F2,B1,CA,D0,FA
     ,84,57,68,68,4C,8B,B1,A6,D3, 2891   <197>
1018 DATA E0,08,B0,03,4C,92,B0,20,D2,FF,A6
     ,58,E0,02,90,09,C6,59,20,D2, 2468   <049>
1019 DATA FF,C6,58,D0,F9,4C,8E,B0,48,4A,4A
     ,4A,4A,20,59,B1,68,29,0F,C9, 2419   <035>
1020 DATA 0A,90,02,69,06,69,30,4C,D2,FF,A2
     ,FC,9A,20,D1,B1,20,48,B2,20, 2261   <073>
1021 DATA EA,B1,20,9F,B2,A5,FC,20,4E,B1,A5
     ,FB,20,4E,B1,20,ED,B1,A9,3A, 2860   <148>
1022 DATA A0,20,20,F2,B1,A9,00,85,59,20,8E
     ,B0,20,ED,B1,A4,59,20,EF,B0, 2530   <233>
1023 DATA 91,FB,C8,84,59,C0,08,90,EC,20,10
     ,B2,A9,12,20,D2,FF,20,8E,B0, 2657   <105>
1024 DATA 20,EF,B0,C5,FF,F0,0D,20,43,B3,A9
     ,14,A0,14,20,F2,B1,4C,A2,B1, 2665   <034>
1025 DATA A9,92,20,D2,FF,20,33,B2,20,E0,B2
     ,20,3F,B2,90,9F,4C,8B,B5,A9, 2648   <123>
1026 DATA 93,20,D2,FF,A2,00,A9,03,9D,00,D8
     ,9D,00,D9,9D,00,DA,9D,00,DB, 2476   <237>
1027 DATA E8,D0,EF,60,A9,0D,2C,A9,20,4C,D2
     ,FF,20,D2,FF,98,4C,D2,FF,20, 2965   <160>
1028 DATA E4,FF,F0,FB,60,84,5D,85,5C,A0,00
     ,B1,5C,F0,06,20,D2,FF,C8,D0, 3100   <077>
1029 DATA F6,60,A5,FB,85,5A,A0,00,84,5B,B1
     ,FB,18,65,5A,85,5A,90,02,E6, 2606   <156>
1030 DATA 5B,06,5A,26,5B,C8,C0,08,90,EC,A5
     ,5A,65,5B,85,FF,60,18,A5,FB, 2467   <219>
1031 DATA 69,08,85,FB,90,02,E6,FC,60,A5,FB
     ,C5,5F,A5,FC,E5,60,60,A0,B3, 3106   <183>
1032 DATA A9,FB,20,FF,B1,A0,01,B9,00,02,20
     ,D2,FF,CC,00,02,C8,90,F4,A9, 2692   <098>
1033 DATA 10,ED,00,02,AA,20,ED,B1,CA,D0,FA
     ,A5,62,20,4E,B1,A5,61,20,4E, 2453   <236>
1034 DATA B1,20,ED,B1,A5,60,20,4E,B1,A5,5F
     ,20,4E,B1,A9,9F,20,D2,FF,20, 2575   <038>
1035 DATA EA,B1,24,5E,10,01,60,A9,12,20,D2
     ,FF,A2,28,20,ED,B1,CA,D0,FA, 2646   <161>
1036 DATA A9,92,4C,D2,FF,A5,D6,C9,16,B0,01
     ,60,A9,A0,85,A4,A9,78,85,A6, 2945   <204>
1037 DATA A9,04,85,A5,85,A7,A2,13,A0,27,B1
     ,A4,91,A6,88,10,F9,CA,F0,19, 2671   <208>
1038 DATA 18,A5,A4,69,28,85,A4,90,02,E6,A5
     ,18,A5,A6,69,28,85,A6,90,E0, 2503   <251>
1039 DATA E6,A7,4C,B6,B2,A9,91,4C,D2,FF,A9
     ,0F,8D,18,D4,A9,00,8D,05,D4, 2776   <000>
1040 DATA A9,F7,8D,06,D4,A9,11,8D,04,D4,A9
     ,32,8D,01,D4,A9,00,8D,00,D4, 2413   <126>
1041 DATA A0,80,20,09,B3,A9,10,8D,04,D4,60
     ,A2,FF,CA,D0,FD,88,D0,F8,60, 2914   <240>
1042 DATA A9,0F,8D,18,D4,A9,2D,8D,05,D4,A9
     ,A5,8D,06,D4,A9,21,8D,04,D4, 2385   <119>
1043 DATA A9,07,8D,01,D4,A9,05,8D,00,D4,A0
     ,FF,20,09,B3,A9,20,8D,04,D4, 2250   <078>
1044 DATA A9,00,8D,01,D4,8D,00,D4,60,38,20
     ,F0,FF,8A,48,98,48,18,A0,06, 2179   <175>
1045 DATA A2,18,20,F0,FF,A0,B4,A9,0A,20,FF
     ,B1,20,12,B3,20,E4,FF,F0,FB, 2931   <093>
1046 DATA A2,1D,A9,14,20,D2,FF,CA,D0,FA,68
     ,A8,68,AA,18,4C,F0,FF,0D,0D, 2704   <088>
1047 DATA 0D,20,20,20,20,20,20,20,4D,41,53
     ,43,48,49,4E,45,4E,53,50,52, 1144   <216>
1048 DATA 41,43,48,45,20,2D,20,45,44,49,54
     ,4F,52,20,0D,0D,20,20,20,20, 1023   <038>
1049 DATA 20,20,20,20,56,4F,4E,20,4E,2E,4D
     ,41,4E,4E,20,26,20,44,2E,57, 1128   <206>
1050 DATA 45,49,4E,45,43,4B,00,0D,0D,0D,20
     ,20,20,50,52,4F,47,52,41,4D, 1102   <117>
1051 DATA 4D,4E,41,4D,45,20,3A,20,00,0D,0D
     ,20,20,20,53,54,41,52,54,41, 1073   <095>
1052 DATA 44,52,45,53,53,45,20,3A,20,24,00
     ,0D,0D,20,20,20,45,4E,44,41, 1014   <129>
1053 DATA 44,52,45,53,53,45,20,20,20,3A,20
     ,24,00,92,05,20,50,52,4F,47, 1171   <217>
1054 DATA 52,41,4D,4D,20,3A,20,00,12,20,20
     ,2A,2A,2A,20,46,41,4C,53,43, 1024   <027>
1055 DATA 48,45,20,45,49,4E,47,41,42,45,20
     ,2A,2A,2A,20,20,92,00,0D,0D, 1058   <098>
1056 DATA 2A,2A,2A,20,45,4E,44,45,20,2A,2A
     ,2A,00,13,05,20,20,12,44,92, 920    <148>
1057 DATA 49,53,4B,20,4F,44,45,52,20,12,54
     ,92,41,50,45,0D,00,13,20,20, 1151   <035>
1058 DATA 49,2F,4F,20,2D,20,46,45,48,4C,45
     ,52,00,20,D1,B1,20,48,B2,A0, 1606   <012>
1059 DATA B3,A9,CF,20,FF,B1,20,8E,B4,85,FC
     ,20,8E,B4,85,FB,C5,61,A5,FC, 3207   <251>
1060 DATA E5,62,90,23,A5,FB,C5,5F,A5,FC,E5
     ,60,B0,19,20,A7,B4,D0,14,60, 2860   <112>
1061 DATA 20,A7,B4,F0,0C,85,F9,20,A7,B4,F0
     ,05,85,F8,4C,EF,B0,68,68,20, 2749   <088>
1062 DATA 43,B3,4C,5F,B4,20,CF,FF,C9,4C,D0
     ,09,20,D1,B1,20,48,B2,4C,0B, 2372   <046>
1063 DATA B6,C9,0D,60,A9,00,85,5E,20,5F,B4
     ,20,EA,B1,20,0D,B5,24,5E,30, 2042   <120>
1064 DATA 05,20,E4,FF,F0,FB,20,E1,FF,F0,26
     ,20,9F,B2,24,5E,10,09,20,4E, 2435   <198>
1065 DATA B5,20,0D,B5,20,60,B5,20,33,B2,20
     ,3F,B2,90,D7,A0,B4,A9,28,20, 2190   <207>
1066 DATA FF,B1,20,E4,FF,C9,0D,D0,F9,A9,00
     ,85,5E,A5,61,85,FB,A5,62,85, 3056   <240>
1067 DATA FC,20,E0,B2,4C,64,B1,A5,FC,20,4E
     ,B1,A5,FB,85,FF,20,4E,B1,A9, 3003   <221>
1068 DATA 20,A0,3A,20,F2,B1,A0,00,20,ED,B1
     ,B1,FB,20,4E,B1,C8,C0,08,90, 2566   <070>
1069 DATA F3,20,ED,B1,24,5E,30,03,A9,12,2C
     ,A9,20,20,D2,FF,20,10,B2,A5, 2190   <059>
1070 DATA FF,20,4E,B1,A9,92,20,D2,FF,4C,EA
     ,B1,A9,FF,85,B8,85,B9,A9,04, 3073   <029>
1071 DATA 85,BA,20,C0,FF,A2,FF,4C,C9,FF,20
     ,CC,FF,A9,FF,4C,C3,FF,20,5F, 3315   <189>
1072 DATA B4,A9,80,85,5E,20,4E,B5,20,48,B2
     ,A2,24,A9,2D,20,D2,FF,CA,D0, 2596   <111>
1073 DATA FA,20,EA,B1,20,EA,B1,20,60,B5,4C
     ,C1,B4,20,B8,B5,A6,5F,A4,60, 2812   <015>
1074 DATA A9,61,20,D8,FF,B0,0A,20,B7,FF,29
     ,BF,D0,03,4C,FB,B4,A9,01,20, 2577   <201>
1075 DATA C3,FF,20,68,B6,A0,B4,A9,4F,20,FF
     ,B1,20,F9,B1,4C,FB,B4,20,68, 2921   <237>
1076 DATA B6,A9,37,A0,B4,20,FF,B1,20,F9,B1
     ,A2,08,C9,44,F0,06,A2,01,C9, 2717   <213>
1077 DATA 54,D0,F1,A9,01,A8,20,BA,FF,A0,00
     ,E0,01,F0,1A,A9,40,8D,20,02, 2403   <101>
1078 DATA A9,3A,8D,21,02,B9,01,02,99,22,02
     ,C8,CC,00,02,90,F4,C8,C8,D0, 2182   <127>
1079 DATA 0C,B9,01,02,99,20,02,C8,CC,00,02
     ,D0,F4,98,A2,20,A0,02,4C,BD, 2018   <025>
1080 DATA FF,20,B8,B5,A5,BA,C9,08,90,33,A6
     ,B9,86,57,A9,01,20,C3,FF,A9, 2800   <022>
1081 DATA 60,85,B9,20,C0,FF,B0,28,A5,BA,20
     ,B4,FF,A5,B9,20,96,FF,20,A5, 2911   <053>
1082 DATA FF,85,61,A5,90,4A,4A,B0,13,20,A5
     ,FF,85,62,20,AB,FF,A5,57,85, 2663   <214>
1083 DATA B9,A9,00,20,D5,FF,90,03,4C,A3,B5
     ,86,5F,84,60,A5,BA,C9,01,D0, 2639   <131>
1084 DATA 0A,AD,3D,03,85,61,AD,3E,03,85,62
     ,4C,FB,B4,A9,13,20,D2,FF,A2, 2300   <120>
1085 DATA 1C,20,ED,B1,CA,D0,FA,60, 1230  <214>
```

© 64'er

MSE (Schluß). Dieses Listing können Sie (müssen aber nicht) mit dem Checksummer 64 V3 in diesem Heft eingeben

Vielstimmig

Das Besondere an diesem Synthesizer ist die Tastaturabfrage: Sie ermöglicht es, bis zu drei verschiedene Töne gleichzeitig zu spielen.

Haben Sie schon einmal versucht, einen Synthesizer in Basic zu programmieren? Dann haben Sie sicher festgestellt, daß die Tastaturabfrage (welcher Ton wird gerade gespielt?) über »GET« oder »PEEK (197)« beim gleichzeitigen Drücken von zwei oder mehr Tasten diese nicht mehr unterscheiden kann. Eigentlich schade, weil dadurch die drei Stimmen des C 64 nicht ausgenützt werden können.

Als ich nun im Buch »PEEKs und POKEs zum C 64« von Data Becker die Möglichkeit entdeckte, mehrere Tasten gleichzeitig abzufragen, kam mir die Idee, eine mehrstimmige Orgel zu programmieren.

Hinweise zum Abtippen

Der Synthesizer besteht aus drei Programmen:
Dem Hauptprogramm »SYNTHESIZER« (Listing 1) und den beiden Maschinensprache-Unterroutinen » SY« (Listing 2) und »TASTEN« (Listing 3).

Das Programm »SYNTHESIZER« ist mit Hilfe des Checksummers einzutippen und erst einmal zu speichern. Sind alle drei Programme auf Diskette/Kassette vorhanden, ist der Synthesizer startbereit. Achten Sie bei den MSE-Listings bitte auf das Leerzeichen vor dem ersten Buchstaben des Programmnamens.

Wer nur eine Datasette zur Verfügung hat, muß beim Speichern auf die Reihenfolge der Programme auf der Kassette achten: Zuerst »SYNTHESIZER«, danach »SY« und schließlich » TASTEN«. Außerdem ist für Datasette in den beiden ersten Programmzeilen des Basic-Programms das »,8,1« in »,1,1« zu ändern.

Bedienung des Synthesizers

Nach dem Start des Basic-Programms mit »RUN« lädt dieses selbst die beiden Maschinenroutinen nach.
Danach kann man den Synthesizer stimmen: F1 und F7 in großen Schritten; F3 und F5 in kleinen. Durch Drücken von Return gelangt man ins Hauptmenü: Man hat jetzt die Wahl zwischen den Funktionen »Wellenform«, »Pulsbreite«, »Filter«, »Lautstärke», »ADSR-Hüllkurve», »Spielen« und »Ende«. Es lassen sich immer nur alle drei Stimmen des C 64 gleichzeitig ändern (dies ist sinnvoll, da beim Spielen zum Beispiel nicht immer der höchste Ton der ersten Stimme entspricht).

Gespielt wird über die beiden Tastenreihen »Z« bis »l« und »Q« bis »↑«. Die Halbtöne liegen auf den entsprechenden Tasten darüber.

Wer sich mit dem Programm genauer beschäftigen möchte, findet in der Tabelle die Variablenbelegung.

Übrigens: Beim reinen, einstimmigen Melodie-Spiel erreicht man einen sehr guten Echo-Effekt, indem man eine Taste so lange hält, bis der nächste Ton der Melodie durch die jeweilige Taste erklingt.

(Georg Gerber/tr)

Verwendete Variablen:

FF	Filterfrequenz
P	Pulsbreite
RZ	Resonanz
AR	Attack
DE	Decay
SU	Sustain
RE	Release
FA	Filterart
L	Lautstärke
F1/2/3	Hoch-/Band-/Tiefpaß (hat den Wert 0 oder 1)
W1/2/3/4/5	Wellenform (nur 0 oder 1)
M	Maximaler Wert beim Unterprogramm »Parameter einstellen«
G	Parameter, der in diesem Menü eingestellt wird
LB/HB	Low-/High-Byte
SI	Startadresse des SID

```
0 :POKE 2053,143:LOAD" SY",8,1::                   <145>
1 :POKE 2081,143:LOAD" TASTEN",8,1                 <113>
2 :                                                <234>
3 REM  ********************************            <053>
4 REM  ********************************            <054>
5 REM  **                            **            <005>
6 REM  **  SYNTHESIZER (2.FASSUNG)   **            <239>
7 REM  **  ===========               **            <069>
8 REM  **                            **            <008>
9 REM  **     GEORG GERBER OKT. 1985 **            <134>
10 REM **                            **            <010>
11 REM **     7500 KARLSRUHE 51      **            <102>
12 REM **                            **            <012>
13 REM **     TULPENSTR.10           **            <206>
14 REM **                            **            <014>
15 REM **     TEL.: 0721/31273       **            <250>
16 REM **                            **            <016>
17 REM **                            **            <017>
20 REM ********************************            <070>
21 REM ********************************            <071>
25 :                                               <001>
32 REM *** PARAMETER SETZTEN ------------          <249>
33 FF=1000:P=2048:RZ=7:AT=0:DE=0:SU=15:RE=
   11:FA=0:L=15:W3=1:F2=1                          <229>
34 GOSUB 20000:REM BILDSCHIRM LOESCHEN             <017>
35 :                                               <011>
36 REM *** ADRESSEN SETZEN -------------           <114>
37 SI=54272:REM * BASISADRESSE 'SID'               <071>
38 POKE SI+24,15:POKE SI+23,0:POKE SI+4,0:
   POKE SI+11,0:POKE SI+18,0                       <132>
39 POKE SI+5,0:POKE SI+6,251:POKE SI+12,0:
   POKE SI+13,251:POKE SI+19,0:POKE SI+20,
   251                                             <018>
40 A=W1*128+W2*64+W3*32+W4*16+W5*8                 <077>
41 POKE 8*4096+11*256,A                            <062>
42 POKE SI+3,8:POKE SI+10,8:POKE SI+17,8           <045>
43 :                                               <019>
44 REM *** GRUNDTON STIMMEN ------------           <208>
45 PRINT"{DOWN}GRUNDTON STIMMEN"                   <019>
46 PRINT"EEEEEEEEEEEEEEEE"                         <191>
47 PRINT"{3DOWN}F1++ / F3+ / F5- / F7 --"          <065>
48 PRINT"{2DOWN,SPACE}<RETURN> : WEITER"           <133>
49 POKE SI+4,33                                    <206>
50 G=2228:W=2↑(1/12)                               <130>
60 A=G:GOSUB 20500:POKE SI,LB:POKE SI+1,HB         <217>
70 IF PEEK(197)=3 THEN G=G-100:IF G<350 TH
   EN G=350                                        <204>
71 IF PEEK(197)=6 THEN G=G-1:IF G<350 THEN
    G=350                                          <051>
72 IF PEEK(197)=5 THEN G=G+1:IF G>6501 THE
   N G=6501                                        <222>
73 IF PEEK(197)=4 THEN G=G+100:IF G>6501 T
   HEN G=6501                                      <011>
75 IF PEEK(197)=1 THEN 80                          <056>
77 GOTO 60                                         <039>
80 POKE SI+4,0:PRINT"{DOWN}MOMENT BITTE"           <221>
90 REM *** TONLEITER BERECHNEN **                  <206>
100 FOR I=0 TO 40:A=G*W↑I:GOSUB 20500:POKE
    8*4096+5*256+9+2*I,LB                          <231>
110 POKE 8*4096+5*256+10+2*I,HB:NEXT               <162>
```

Listing 1. Zur Eingabe von »SYNTHESIZER« verwenden Sie bitte den Checksummer (Seite 6)

64er-online.de

```
120 POKE 198,0                                   <028>
123 :                                            <099>
200 REM *** HAUPTMENU ------------------         <205>
210 GOSUB 20000:PRINT"{2DOWN}"                   <152>
220 PRINT"W ... WELLENFORM"                      <108>
221 PRINT"P ... PULSBREITE"                      <078>
222 PRINT"F ... FILTER"                          <241>
223 PRINT"L ... LAUTSTAERKE"                     <005>
224 PRINT"A ... ADSR-HUELLKURVE"                 <088>
225 PRINT"S ... SPIELEN"                         <070>
226 PRINT"E ... ENDE"                            <082>
230 PRINT"{DOWN,SPACE}BITTE WAEHLEN"             <027>
240 GET A$:IF A$=""THEN 240                      <146>
250 IF A$="W"THEN 300                            <190>
251 IF A$="P"THEN 500                            <068>
252 IF A$="F"THEN 700                            <072>
253 IF A$="L"THEN 900                            <084>
254 IF A$="A"THEN 1100                           <189>
255 IF A$="S"THEN 1300                           <215>
256 IF A$="E"THEN 1500                           <227>
260 GOTO 240                                     <006>
266 :                                            <244>
300 REM *** WELLENFORM -----------------         <050>
310 GOSUB 20000:PRINT"{DOWN,7SPACE}WELLENF
    ORM"                                         <201>
320 PRINT"{DOWN}RAUSCHEN{2SPACE}";W1             <004>
321 PRINT"RECHTECK{2SPACE}";W2                   <055>
322 PRINT"SAEGEZAHN{2SPACE}";W3                  <032>
323 PRINT"DREIECK{2SPACE}";W4                    <242>
324 PRINT"TEST{2SPACE}";W5                       <070>
330 PRINT"{6UP}"                                 <118>
339 INPUT"{9RIGHT}";A$:IF VAL(A$)=0 AND A$
    <>"0"THEN PRINT"{2UP}":GOTO 339              <049>
340 W1=VAL(A$):IF W1<>1 AND W1<>0 THEN PRI
    NT"{2UP}":GOTO 339                           <096>
341 INPUT"{9RIGHT}";A$:IF VAL(A$)=0 AND A$
    <>"0"THEN PRINT"{2UP}":GOTO 341              <187>
342 W2=VAL(A$):IF W2<>1 AND W2<>0 THEN PRI
    NT"{2UP}":GOTO 341                           <131>
343 INPUT"{10RIGHT}";A$:IF VAL(A$)=0 AND A
    $<>"0"THEN PRINT"{2UP}":GOTO 343             <026>
344 W3=VAL(A$):IF W3<>1 AND W3<>0 THEN PRI
    NT"{2UP}":GOTO 343                           <185>
345 INPUT"{8RIGHT}";A$:IF VAL(A$)=0 AND A$
    <>"0"THEN PRINT"{2UP}":GOTO 345              <083>
346 W4=VAL(A$):IF W4<>1 AND W4<>0 THEN PRI
    NT"{2UP}":GOTO 345                           <239>
347 INPUT"{5RIGHT}";A$:IF VAL(A$)=0 AND A$
    <>"0"THEN PRINT"{2UP}":GOTO 347              <155>
348 W5=VAL(A$):IF W5<>1 AND W5<>0 THEN PRI
    NT"{2UP}":GOTO 347                           <037>
350 A=W1*128+W2*64+W3*32+W4*16+W5*8              <133>
360 POKE 8*4096+11*256,A                         <127>
370 GOTO 200                                     <052>
377 :                                            <099>
500 REM *** PULSBREITE -----------------         <016>
510 GOSUB 20000:PRINT"{DOWN,9SPACE}PULSBRE
    ITE"                                         <150>
520 G=P:M=4095:GOSUB 21000:P=G                   <156>
530 A=G:GOSUB 20500:POKE SI+2,LB:POKE SI+3
    ,HB:POKE SI+9,LB:POKE SI+10,HB               <105>
531 POKE SI+16,LB:POKE SI+17,HB                  <060>
540 GOTO 200                                     <224>
544 :                                            <118>
700 REM *** FILTER ---------------------         <131>
710 GOSUB 20000:PRINT"{DOWN,7SPACE}FILTER"       <118>
720 PRINT"FREQUENZ":G=FF:M=2047:GOSUB 2100
    0:FF=G                                       <193>
730 PRINT"FILTER{SPACE,RVSON}A{RVOFF}N / A
    {RVSON}U{RVOFF}S"                            <121>
735 GET A$:IF A$<>"A"AND A$<>"U"THEN 735         <223>
740 MO=7:IF A$="U"THEN MO=0                      <251>
750 PRINT"{DOWN}RESONANZ{DOWN}":G=RZ:M=15:
    GOSUB 21000:RZ=G                             <224>
760 POKE SI+23,G*16+MO                           <169>
780 PRINT"{2DOWN}ART:{DOWN}"                     <082>
800 PRINT"TIEF ";F1                              <201>
801 PRINT"BAND ";F2                              <225>
802 PRINT"HOCH ";F3                              <074>
803 PRINT"{4UP}":POKE 198,0                      <081>
810 INPUT"{4RIGHT}";A$:IF A$<>"0"AND VAL(A
    $)=0 THEN PRINT"{2UP}":GOTO 810              <207>
811 F1=VAL(A$):IF F1<>1 AND F1<>0 THEN PRI
    NT"{2UP}":GOTO 810                           <017>
812 INPUT"{4RIGHT}";A$:IF A$<>"0"AND VAL(A
    $)=0 THEN PRINT"{2UP}":GOTO 812              <210>
813 F2=VAL(A$):IF F2<>1 AND F2<>0 THEN PRI
    NT"{2UP}":GOTO 812                           <071>
814 INPUT"{4RIGHT}";A$:IF A$<>"0"AND VAL(A
    $)=0 THEN PRINT"{2UP}":GOTO 814              <213>
815 F3=VAL(A$):IF F3<>1 AND F3<>0 THEN PRI
    NT"{2UP}":GOTO 814                           <125>
820 FA=F1*16+F2*32+F3*64                         <088>
830 POKE SI+24,FA+L:GOTO 200                     <243>
888 :                                            <102>
900 REM *** LAUTSTAERKE ----------------         <209>
910 GOSUB 20000:PRINT"{DOWN,9SPACE}LAUTSTA
    ERKE{DOWN}"                                  <035>
920 G=L:M=15:GOSUB 21000:L=G:POKE SI+24,L+
    FA:GOTO 200                                  <170>
999 :                                            <213>
1100 REM *** ADSR ----------------------         <160>
1110 GOSUB 20000:PRINT"{4SPACE}ADSR-HUELLK
     URVE{DOWN}"                                 <114>
1120 PRINT"ATTACK":G=AT:M=15:GOSUB 21000:A
     T=G:POKE 198,0                              <079>
1130 PRINT"{2DOWN}DECAY":G=DE:M=15:GOSUB 2
     1000:DE=G:POKE 198,0                        <055>
1140 PRINT"{2DOWN}SUSTAIN":G=SU:M=15:GOSUB
      21000:SU=G:POKE 198,0                      <123>
1150 PRINT"{2DOWN}RELEASE":G=RE:M=15:GOSUB
      21000:RE=G:POKE 198,0                      <117>
1180 A=AT*16+DE:POKE SI+5,A:POKE SI+12,A:P
     OKE SI+19,A                                 <128>
1185 A=SU*16+RE:POKE SI+6,A:POKE SI+13,A:P
     OKE SI+20,A                                 <192>
1190 GOTO 200                                    <110>
1195 :                                           <155>
1300 REM *** SPIELEN -------------------         <032>
1310 GOSUB 20000:PRINT"{8SPACE}SPIELEN"          <185>
1315 POKE 53281,4                                <231>
1320 PRINT"{3DOWN}F7:ENDE{3DOWN}"                <130>
1327 PRINT"{BLUE}TTTTTTTTTTTTTTTTTTTTTTTTT
     TTTTTTTTTTTTTTT";                           <164>
1330 PRINT"{RVSON,WHITE}Q{BLACK}2{WHITE}W{
     BLACK}3{WHITE}ER{BLACK}5{WHITE}T{BLAC
     K}6{WHITE}Y{BLACK}7{WHITE}UI{BLACK}9{
     WHITE}O{BLACK}0{WHITE}P@{BLACK}-{WHIT
     E}*{BLACK}£{WHITE}↑{BLACK}C{WHITE}Z";      <174>
1331 PRINT"{RVSON,WHITE}X{BLACK}D{WHITE}C{
     BLACK}F{WHITE}VB{BLACK}H{WHITE}N{BLAC
     K}J{WHITE}M{BLACK}K{WHITE}<>{BLACK}:{
     WHITE}/{BLACK};{WHITE}";                    <088>
1340 PRINT"{RVSON,WHITE,SPACE,BLACK,SPACE,
     WHITE,SPACE,BLACK,SPACE,WHITE,SPACE}G
     {BLACK,SPACE,WHITE,SPACE,BLACK,SPACE,
     WHITE,SPACE,BLACK,SPACE,WHITE,SPACE}G
     {BLACK,SPACE,WHITE,SPACE,BLACK,SPACE,
     WHITE,SPACE}G{BLACK,SPACE,WHITE,SPACE
     ,BLACK,SPACE,WHITE,SPACE,BLACK,SPACE,
     WHITE,SPACE}";                              <247>
1341 PRINT"{RVSON,WHITE}H{BLACK,SPACE,WHIT
     E,SPACE,BLACK,SPACE,WHITE,SPACE}G{BLA
     CK,SPACE,WHITE,SPACE,BLACK,SPACE,WHIT
     E,SPACE,BLACK,SPACE,WHITE,SPACE}G{BLA
     CK,SPACE,WHITE,SPACE,BLACK,SPACE,WHIT
     E}";                                        <077>
1350 PRINT"{RVSON,WHITE,SPACE}B B GB B B G
     B B GB B B ";                               <046>
1351 PRINT"{RVSON,WHITE}HB B HB B B HB B";       <206>
1355 PRINT"{BLUE,RVOFF}UUUUUUUUUUUUUUUUUUU
     UUUUUUUUUUUUUUUUUUUUU"                      <131>
1390 SYS 8*4096:POKE 198,0:GOTO 200              <146>
1440 :                                           <146>
1500 REM *** ENDE ----------------------         <015>
1510 GOSUB 20000:PRINT"{3DOWN}AUF WIEDERSE
     HEN{3DOWN,LIG.BLUE}"                        <060>
1520 SYS 42115:REM * END OHNE READY              <015>
19997 :                                          <161>
19998 :                                          <162>
19999 :                                          <163>
20000 REM *** BILDSCHIRM ---------------         <223>
20005 POKE 53281,6                               <127>
20010 PRINT"{CLR,DOWN,WHITE,8SPACE}POLYPHO
      NER SYNTHESIZER"                           <078>
20011 PRINT"{7SPACE}=====================
      =={DOWN}"                                  <132>
20012 PRINT"{20SPACE}<C> GEORG GERBER '85"       <176>
20020 RETURN                                     <010>
20025 :                                          <189>
```

Listing 1. »SYNTHESIZER« (Fortsetzung)

64er-online.de
64er-online.net

```
20500 REM *** LB / HB AUS A BERECHNEN --       <196>
20510 HB=INT(A/256):LB=A-HB*256:RETURN         <006>
20560 :                                        <216>
21000 REM *** PARAMETER EINSTELLEN -----       <206>
21002 PRINT"F1 ++ / F3 + / F5 - / F7 -- "      <161>
21003 PRINT"<RETURN> : ENDE{2DOWN}"            <141>
21010 PRINT"{UP,5SPACE,5LEFT}";G               <219>
21020 IF PEEK(197)=3 THEN G=G-100:IF G<0 T
      HEN G=0                                  <226>
21030 IF PEEK(197)=6 THEN G=G-1:IF G<0 THE
      N G=0                                    <158>
21040 IF PEEK(197)=5 THEN G=G+1:IF G>M THE
      N G=M                                    <225>
21050 IF PEEK(197)=4 THEN G=G+100:IF G>M T
      HEN G=M                                  <123>
21060 IF PEEK(197)=1 THEN RETURN               <052>
21070 GOTO 21010                               <038>
```

© 64'er

Listing 1. »SYNTHESIZER« (Schluß)

```
 programm :  sy                8000 81c2
----------------------------------------

8000 : ad 0e dc 29 fe 8d 0e dc  5f
8008 : a9 00 8d 00 85 8d 01 85  e9
8010 : 8d 02 85 8d 03 85 8d 04  4c
8018 : 85 8d 05 85 a0 00 20 78  d1
8020 : 81 b9 00 8a 8d 00 dc ad  77
8028 : 01 dc 49 ff 39 01 8a d9  63
8030 : 01 8a d0 4c ad 00 85 d9  d9
8038 : 09 85 d0 0b ad 01 85 d9  46
8040 : 0a 85 d0 03 4c 7d 80 ad  af
8048 : 02 85 d9 09 85 d0 0b ad  0b
8050 : 03 85 d9 0a 85 d0 03 4c  51
8058 : 7d 80 ad 04 85 d9 09 85  58
8060 : d0 0b ad 05 85 d9 0a 85  1c
8068 : d0 03 4c 7d 80 ae 08 85  25
8070 : b9 09 85 9d 00 85 e8 b9  06
8078 : 0a 85 9d 00 85 4c c9 80  8f
8080 : ea ad 00 85 d9 09 85 d0  8f
8088 : 10 ad 01 85 d9 0a 85 d0  05
8090 : 08 a9 00 8d 00 85 8d 01  83
8098 : 85 ad 02 85 d9 09 85 d0  c3
80a0 : 10 ad 03 85 d9 0a 85 d0  9e
80a8 : 08 a9 00 8d 02 85 8d 03  bf
80b0 : 85 ad 04 85 d9 09 85 d0  5b
80b8 : 10 ad 05 85 d9 0a 85 d0  36
80c0 : 08 a9 00 8d 04 85 8d 05  fb
80c8 : 85 ea c8 c8 c0 52 f0 03  76
80d0 : 4c 1e 80 ad 00 85 c9 00  54
80d8 : d0 12 ad 01 85 c9 00 d0  85
80e0 : 0b ad 00 8b 29 fe 8d 04  fc
80e8 : d4 4c 00 81 ad 00 85 8d  1f
80f0 : 00 d4 ad 01 85 8d 01 d4  58
80f8 : ad 00 8b 09 01 8d 04 d4  e0
8100 : ad 02 85 c9 00 d0 12 ad  73
8108 : 03 85 c9 00 d0 0b ad 00  5c
8110 : 8b 29 fe 8d 0b d4 4c 2d  84
8118 : 81 ad 02 85 8d 07 d4 ad  61
8120 : 03 85 8d 08 d4 ad 00 8b  1c
8128 : 09 01 8d 0b d4 ad 04 85  4c
8130 : c9 00 d0 12 ad 05 85 c9  1c
8138 : 00 d0 0b ad 00 8b 29 fe  18
8140 : 8d 12 d4 4c 5a 81 ad 04  05
8148 : 85 8d 0e d4 ad 05 85 8d  e6
8150 : 0f d4 ad 00 8b 09 01 8d  55
8158 : 12 d4 ea a9 fe 8d 00 dc  da
8160 : ad 01 dc 49 ff 29 08 c9  eb
8168 : 08 d0 09 ad 0e dc 09 01  be
8170 : 8d 0e dc 60 4c 1c 80 60  b0
8178 : ea ad 00 85 c9 00 d0 0d  e4
8180 : ad 01 85 c9 00 d0 06 a9  3a
8188 : 00 8d 08 85 60 ad 02 85  88
8190 : c9 00 d0 0d ad 03 85 c9  cc
8198 : 00 d0 06 a9 02 8d 08 85  6f
81a0 : 60 ad 04 85 c9 00 d0 0d  83
81a8 : ad 05 85 c9 00 d0 06 a9  64
81b0 : 04 8d 08 85 60 a9 06 8d  b4
81b8 : 08 85 60 60 a9 06 8d 08  b8
81c0 : 85 60 04 00 98 98 b8 a8  f9
```

Listing 2. »SY«, die Tastaturabfrage zum Synthesizer. Eingabe mit dem MSE. Achten Sie auch auf das Leerzeichen vor dem »SY«.

```
 programm :  tasten            8a00 8a56
----------------------------------------

8a00 : 7f 40 7f 08 fd 02 fd 01  6a
8a08 : fd 40 fb 02 fb 01 fb 40  9d
8a10 : fb 08 f7 02 f7 01 f7 40  35
8a18 : ef 02 ef 01 ef 40 ef 08  f5
8a20 : df 02 df 40 df 08 bf 02  41
8a28 : bf 01 bf 40 bf 08 fd 10  b4
8a30 : fb 80 fb 04 fb 10 fb 20  5b
8a38 : f7 80 f7 10 f7 20 ef 80  b0
8a40 : ef 04 ef 10 ef 20 df 80  b0
8a48 : df 10 df 20 bf 80 bf 04  32
8a50 : bf 20 00 00 00 0d 55 05  e7
```

Listing 3. » TASTEN«. Verwenden Sie zur Eingabe bitte den MSE (Seite 7). Achten Sie auch hier auf das Leerzeichen vor dem »TASTEN«.

Elektronisches Akkordeon

Mit diesem Programm steht dem Einzug des C 64 in die Hausmusik nichts mehr im Wege. Spielen Sie auf der Tastatur wie auf einem richtigen Akkordeon.

Wer sich das Listing genauer betrachtet, wird bemerken, wie einfach doch das Programm aufgebaut ist. Trotzdem produziert es erstaunlich harmonische Klänge. Außerdem wurden die einzelnen Akkorde so auf der Tastatur verteilt, daß geübte Akkordeonspieler nach einer kurzen Eingewöhnungsphase wie auf einem »echten« Instrument spielen können.

Wenn das Programm mit »RUN« gestartet wurde, befindet man sich sofort im Spielmodus. Wie die Tastatur mit den Akkorden belegt ist, zeigt unser Bild.

Durch Druck auf die große Space-Taste gelangt man in einen Änderungsmodus. Folgende Funktionen sind über die Zifferntasten verfügbar (jeweils vorher Space-Taste drücken):

»1«: Baß auf Sägezahnschwingung schalten.
»2«: Baß auf Rechteckschwingung schalten.



Das Bild zeigt die Belegung der Tastatur mit den einzelnen Bässen (Großbuchstaben) beziehungsweise Akkorden (Kleinbuchstaben)

»3«: Baß auf Dreieckschwingung schalten.
»4«: Baß auf Rauschen schalten.
»5«: Akkord auf Sägezahnschwingung schalten.
»6«: Akkord auf Rechteckschwingung schalten.
»7«: Akkord auf Dreieckschwingung schalten
»8«: Akkord auf Rauschen schalten.
»9«: Kleine Rechteckimpulsweite einschalten (nur in Verbindung mit Rechteckschwingung).
»0« Große Rechteckschwingung einschalten (nur in Verbindung mit Rechteckschwingung).
»*« Einschaltwerte wiederherstellen (»1«, »5« und »9«).

Es empfiehlt sich übrigens, den Computer mit einem Überspielkabel über die Audio/Video-Buchse an eine Stereoanlage anzuschließen. Man erhält so einen wesentlich besseren Klang, vor allem fülligere Bässe. Auch sollte man ruhig einmal mit der ADSR-Hüllkurve experimentieren. Sie wird in den Zeilen 10 bis 60 festgelegt.

Technisches zum Programm

Das eigentliche Problem bestand darin, die 40 Tasten schnellstmöglich abzufragen. Eine Kolonne von IF-THEN-Anweisungen wäre sicher nicht die Optimallösung. Der Programmautor hatte da folgende geniale Idee: In den Zeilen 80 und 90 wird auf einen Tastendruck gewartet, dieser als Buchstabe beziehungsweise Zahl übernommen und einem Zahlenwert (ASC-Funktion) zugeordnet. Der Wert wird nun in ein anderes Format umgerechnet (Zeilen 300, 320, 340 und 360) und über eine Reihe von ON-GOTO-Befehlen der entsprechende Dreiklang/Baß angesprungen.

Auf eine aufwendige Bildschirmgestaltung wurde zum Wohle des abtippenden Lesers verzichtet.

(Hermann Huck/tr)

```
1 REM  A K K O R D E O N  (2)                <241>
2 REM  VON HERMANN  H U C K                 <244>
3 REM  7987 WEINGARTEN                      <185>
4 REM  LAURASTR.34                          <004>
6 A=54272:B=54273:REM FUER TONHOEHE         <234>
7 C=54279:D=54280:REM FUER TONHOEHE         <128>
8 E=54286:F=54287:REM FUER TONHOEHE         <022>
10 POKE 54277,1*16+15:REM ANSCHL.ABSCHW.    <138>
20 POKE 54284,1*16+15:REM ANSCHL.ABSCHW.    <084>
30 POKE 54291,1*16+15:REM ANSCHL.ABSCHW.    <028>
40 POKE 54278,15*16+8:REM HALT.AUSKL.       <112>
50 POKE 54285,15*16+8:REM HALT.AUSKL.       <057>
60 POKE 54292,15*16+8:REM HALT.AUSKL.       <002>
70 POKE 54296,15:REM LAUTST.                <169>
71 TL=255 :REM TASTENVERH.LOW  GRUNDEIN.    <214>
72 TH=0   :REM TASTENVERH.HIGH GRUNDEIN.    <005>
74 WB=33  :REM WELLENFORM BASS GRUNDEIN.    <182>
76 WA=33  :REM WELLENFORM AKK. GRUNDEIN.    <254>
77 POKE 54274,TL:POKE 54275,TH              <086>
78 POKE 54281,TL:POKE 54282,TH              <017>
79 POKE 54288,TL:POKE 54289,TH              <164>
80 GET T$:IF T$=""THEN 80                   <188>
90 X=ASC(T$)                                <017>
100 IF X<=58 THEN 200                       <154>
110 IF X>=65 THEN 250                       <086>
120 GOTO 80                                 <098>
200 IF X>=44 AND X<=50 THEN 300             <125>
210 IF X>=51 THEN 320                       <110>
215 IF X=32 THEN 5200                       <145>
220 GOTO 80                                 <198>
250 IF X<=90 AND X>=78 THEN 360             <045>
260 IF X<=77 THEN 340                       <077>
270 GOTO 80                                 <250>
300 X=X-43                                  <196>
310 ON X GOTO 2150,80,2210,2250,1750,610,7
    10                                      <188>
320 X=X-50                                  <151>
330 ON X GOTO 1210,1310,1410,1510,1610,165
    0,1710,1550                             <002>
340 X=X-64                                  <237>
350 ON X GOTO 1150,2010,1910,950,910,850,6
    50,750,1310,1250,1350,1450,2110         <234>
360 X=X-77                                  <194>
370 ON X GOTO 2050,1410,1510,1110,810,1050
    ,610,1210,1950,1010,1850,710,1810       <060>
610 POKE A,91 :POKE B,4                     <140>
620 POKE C,90 :POKE D,4                     <106>
630 POKE E,89 :POKE F,4 :REM C              <014>
640 GOTO 5000                               <104>
650 POKE A,103:POKE B,17                    <079>
660 POKE C,237:POKE D,21                    <163>
670 POKE E,20 :POKE F,26:REM C-DUR          <137>
680 GOTO 5100                               <160>
710 POKE A,134:POKE B,6                     <054>
720 POKE C,133:POKE D,6                     <215>
730 POKE E,132:POKE F,6 :REM G              <070>
740 GOTO 5000                               <204>
750 POKE A,109:POKE B,16                    <150>
760 POKE C,137:POKE D,19                    <216>
770 POKE E,20 :POKE F,26:REM G-DUR          <241>
780 GOTO 5100                               <006>
810 POKE A,208:POKE B,5                     <238>
820 POKE C,207:POKE D,5                     <143>
830 POKE E,206:POKE F,5 :REM F              <126>
840 GOTO 5000                               <050>
850 POKE A,103:POKE B,17                    <025>
860 POKE C,59 :POKE D,23                    <064>
870 POKE E,69 :POKE F,29:REM F-DUR          <199>
880 GOTO 5100                               <106>
910 POKE A,194:POKE B,7                     <146>
920 POKE C,193:POKE D,7                     <051>
930 POKE E,192:POKE F,7 :REM B              <032>
940 GOTO 5000                               <150>
950 POKE A,137:POKE B,19                    <128>
960 POKE C,59 :POKE D,23                    <164>
970 POKE E,3  :POKE F,31:REM B-DUR          <189>
980 GOTO 5100                               <206>
1010 POKE A,46 :POKE B,5                    <198>
1020 POKE C,45 :POKE D,5                    <164>
1030 POKE E,44 :POKE F,5 :REM ES            <018>
1040 GOTO 5000                              <252>
1050 POKE A,129:POKE B,15                   <038>
1060 POKE C,178:POKE D,20                   <121>
1070 POKE E,20 :POKE F,26:REM ES-DUR        <015>
1080 GOTO 5100                              <052>
1110 POKE A,233:POKE B,6                    <105>
1120 POKE C,232:POKE D,6                    <011>
1130 POKE E,231:POKE F,6 :REM AS            <201>
1140 GOTO 5000                              <096>
1150 POKE A,103:POKE B,17                   <071>
1160 POKE C,178:POKE D,20                   <221>
1170 POKE E,160:POKE F,27:REM AS-DUR        <076>
1180 GOTO 5100                              <152>
1210 POKE A,227:POKE B,4                    <111>
1220 POKE C,226:POKE D,4                    <017>
1230 POKE E,225:POKE F,4 :REM D             <254>
1240 GOTO 5000                              <196>
1250 POKE A,162:POKE B,14                   <076>
1260 POKE C,137:POKE D,19                   <208>
1270 POKE E,157:POKE F,24:REM D-DUR         <181>
1280 GOTO 5100                              <254>
1310 POKE A,82 :POKE B,7                    <134>
1320 POKE C,81 :POKE D,7                    <100>
1330 POKE E,80 :POKE F,7 :REM A             <038>
1340 GOTO 5000                              <042>
1350 POKE A,162:POKE B,14                   <178>
1360 POKE C,112:POKE D,18                   <020>
1370 POKE E,237:POKE F,21:REM A-DUR         <087>
1380 GOTO 5100                              <098>
1410 POKE A,124:POKE B,5                    <168>
1420 POKE C,123:POKE D,5                    <073>
1430 POKE E,122:POKE F,5 :REM E             <183>
1440 GOTO 5000                              <142>
1450 POKE A,208:POKE B,13                   <152>
1460 POKE C,109:POKE D,16                   <122>
1470 POKE E,237:POKE F,21:REM E-DUR         <191>
1480 GOTO 5100                              <198>
1510 POKE A,56 :POKE B,8                    <246>
1520 POKE C,55 :POKE D,8                    <212>
1530 POKE E,54 :POKE F,8 :REM H             <088>
1540 GOTO 5000                              <244>
1550 POKE A,78 :POKE B,12                   <204>
1560 POKE C,109:POKE D,16                   <224>
1570 POKE E,178:POKE F,20:REM H-DUR         <106>
1580 GOTO 5100                              <044>
1610 POKE A,40 :POKE B,6                    <171>
1620 POKE C,39 :POKE D,6                    <235>
1630 POKE E,38 :POKE F,6 :REM FIS           <229>
1640 GOTO 5000                              <088>
1650 POKE A,157:POKE B,4                    <204>
1660 POKE C,156:POKE D,4                    <110>
1670 POKE E,155:POKE F,4 :REM CIS           <201>
1680 GOTO 5000                              <128>
```

Das Listing zum elektronischen Akkordeon. Verwenden Sie zur Eingabe bitte den Checksummer (Seite 6).

64er-online.de

Immer im Takt

Vollwertige Baß-Begleitung, gespielt von Ihrem C 64. Ohne komplizierte Eingaben. Das Programm »Bassist« errechnet die Schlagzeug- und Baßbegleitung selbst. Fetziger Sound vom Jazz bis hin zur Unterhaltungsmusik ist vorprogrammiert.

Jeder Amateurmusiker wird mit uns einer Meinung sein: Die Klangqualitäten des C 64 qualifizieren ihn als Rhythmus- und Begleitinstrument!
Oft dürfte dies jedoch an der nicht ganz einfachen Programmierung des Sound-Chips gescheitert sein. »Bassist« befreit Sie vom Betriebssystem, indem er die Steuerung des Sound-Chips nach Ihren Eingaben vornimmt. Sie müssen lediglich die im Jazz und in der Unterhaltungsmusik üblichen Harmonie-Bezeichnungen eingeben, der Computer spielt den entsprechenden Baß und, wenn gewünscht, auch die Schlagzeugbegleitung dazu.

Das Programm ist menügesteuert und sagt dem Benutzer immer, was zu tun ist. Den Menüpunkt »Neuerfassen/Editieren« beendet man durch die Eingabe von »*«. Ansonsten kommt man immer mit der F1-Taste zurück ins Hauptmenü.

Programm-Aufbau:

Das Programm besteht aus zwei Teilen: einem Basic-Programm (Listing 1) und einem Maschinensprache-Programm (Listing 2). In diesem werden Interrupt-gesteuert die SID-Register bedient sowie einige zeitkritische Ton-Bestimmungen durchgeführt. Die Funktionsweise dieses Teils können Sie Listing 3, dem ausführlich dokumentierten Quellcode entnehmen.

SID-Tonerzeugungen

Um die Anschlagsdynamik eines Basses zu simulieren, werden der VCO-Nr.1 und VCO-Nr.2 mit unterschiedlichen Decay-Zeiten und unterschiedlichen Wellenformen angesteuert.
Für die Schlagzeugbegleitung wird VCO-Nr.3 benutzt.

Parameter-Einstellungen

Der Charakter eines Stückes wird wesentlich durch diese Einstellungen bestimmt:

1. Stimmung: Die Stimmung kann um fünf Halbtöne in 1/6-Ton-Schritten verändert werden. (Anpassen an andere Instrumente oder Transponieren eines Stückes!)
2. Tempo: Das Tempo ist von 30 bis 180 Schläge in der Minute einstellbar.
3. Baß-Filter: Vom dunklen String-Bass bis zum hellen Elektro-Baß einstellbar.
4. Baß-Noten: Diese Einstellung bestimmt, wie oft 1/4-Noten beziehungsweise 1/2-Noten gespielt werden. Ist der Pfeil ganz links, werden alle 1/4-Noten gespielt. Ganz rechts nur 1/2-Noten. Dazwischen je nach eingestellter Wahrscheinlichkeit.
5. Baß-Linie: Eine gute Baß-Linie springt nicht sinnlos in einer Tonleiter herum. Vielmehr folgt sie einem Trend, der eine Zeitlang beibehalten wird und dann durch eine Wende wieder abgebrochen wird. Nur »Trend-Spiel« wird aber auch langweilig. Deshalb ist eine Mischung von »Trend« und »Zufall« (die hier einstellbar ist) für eine gelungene Baß-Linie nötig. Normalerweise wird man ziemlich viel »Trend« mit einem kleinen Schuß »Zufall« einstellen.

(R. Treichler/og)

```
1710 POKE A,233:POKE B,6                     <197>
1720 POKE C,232:POKE D,6                     <103>
1730 POKE E,231:POKE F,6 :REM GIS            <196>
1740 GOTO 5000                               <188>
1750 POKE A,46 :POKE B,5                     <176>
1760 POKE C,45 :POKE D,5                     <142>
1770 POKE E,44 :POKE F,5 :REM DIS            <008>
1780 GOTO 5000                               <228>
1810 POKE A,109:POKE B,16                    <196>
1820 POKE C,178:POKE D,20                    <119>
1830 POKE E,160:POKE F,27:REM AS-MOLL        <031>
1840 GOTO 5100                               <050>
1850 POKE A,129:POKE B,15                    <076>
1860 POKE C,178:POKE D,20                    <159>
1870 POKE E,157:POKE F,24:REM ES-MOLL        <046>
1880 GOTO 5100                               <090>
1910 POKE A,112:POKE B,18                    <038>
1920 POKE C,59 :POKE D,23                    <108>
1930 POKE E,3  :POKE F,31:REM B -MOLL        <204>
1940 GOTO 5100                               <150>
1950 POKE A,103:POKE B,17                    <109>
1960 POKE C,59 :POKE D,23                    <148>
1970 POKE E,160:POKE F,27:REM F -MOLL        <178>
1980 GOTO 5100                               <190>
2010 POKE A,103:POKE B,17                    <169>
2020 POKE C,178:POKE D,20                    <063>
2030 POKE E,20 :POKE F,26:REM C -MOLL        <113>
2040 GOTO 5100                               <250>
2050 POKE A,129:POKE B,15                    <022>
2060 POKE C,137:POKE D,19                    <248>
2070 POKE E,20 :POKE F,26:REM G -MOLL        <157>
2080 GOTO 5100                               <129>
2110 POKE A,162:POKE B,14                    <176>
2120 POKE C,137:POKE D,19                    <052>
2130 POKE E,59 :POKE F,23:REM D -MOLL        <202>
2140 GOTO 5100                               <096>
2150 POKE A,162:POKE B,14                    <216>
2160 POKE C,103:POKE D,17                    <089>
2170 POKE E,237:POKE F,21:REM A -MOLL        <153>
2180 GOTO 5100                               <136>
2210 POKE A,10 :POKE B,13                    <178>
2220 POKE C,109:POKE D,16                    <120>
2230 POKE E,237:POKE F,21:REM E -MOLL        <217>
2240 GOTO 5100                               <196>
2250 POKE A,78 :POKE B,12                    <140>
2260 POKE C,109:POKE D,16                    <160>
2270 POKE E,137:POKE F,19:REM H -MOLL        <213>
2280 GOTO 5100                               <236>
5000 REM BASS AUSLOESEN                      <204>
5010 POKE 54276,WB                           <019>
5020 POKE 54283,WB                           <220>
5030 POKE 54290,WB:REM EINSCHALTEN           <089>
5040 FOR I=1 TO 100:NEXT:REM KLANGDAUER      <164>
5050 POKE 54276,WB-1                         <171>
5060 POKE 54283,WB-1                         <116>
5070 POKE 54290,WB-1:REM AUSSCHALTEN         <051>
5080 GOTO 80                                 <232>
5100 REM AKKORD AUSLOESEN                    <176>
5110 POKE 54276,WA                           <115>
5120 POKE 54283,WA                           <062>
5130 POKE 54290,WA:REM EINSCHALTEN           <187>
5140 FOR I=1 TO 100:NEXT:REM KLANGDAUER      <010>
5150 POKE 54276,WA-1                         <013>
5160 POKE 54283,WA-1                         <214>
5170 POKE 54290,WA-1:REM AUSSCHALTEN         <149>
5180 GOTO 80                                 <078>
5200 GET T$:IF T$="" THEN 5200               <075>
5210 IF T$="1"THEN WB=33 :GOTO 80            <029>
5220 IF T$="2"THEN WB=65 :GOTO 80            <137>
5230 IF T$="3"THEN WB=17 :GOTO 80            <243>
5240 IF T$="4"THEN WB=129:GOTO 80            <177>
5250 IF T$="5"THEN WA=33 :GOTO 80            <063>
5260 IF T$="6"THEN WA=65 :GOTO 80            <171>
5270 IF T$="7"THEN WA=17 :GOTO 80            <021>
5280 IF T$="8"THEN WA=129:GOTO 80            <211>
5290 IF T$="9"THEN TL=255:TH=0:GOTO 77       <240>
5300 IF T$="0"THEN TL=0  :TH=8:GOTO 77       <154>
5305 IF T$="*"THEN 71                        <188>
5310 GOTO 5200                               <234>
```

© 64'er

Das Listing zum elektronischen Akkordeon (Schluß)

Variable	Bedeutung
A4	Anzahl 1/4-Schläge pro Takt
AP	Adresse des Assembler-Programms
AU	Vergleichswert für HD zum Ausschalten im letzten Takt
AZ	Switch für Anzählen
C$	= CHR$ (13)
DT, DS	Disk-Track/Sektor
ER, ER$, ET, ES	Disk-Error-Variablen
F	aktuelle Frequenz
F()	Frequenz-Tabelle für 1 Oktave
F1,F2	Eck-Frequenzen für Trend-Umkehr
F6	Frequenz-Faktor für plus 1/6-Ton
FA	Frequenz-Faktor für Aufschlag
FK	Frequenz-Konstante für Umrechnung auf SID-Werte
FL	letzte gespielte Frequenz
FO	Frequenz-Faktor für etwa plus 1/2 Oktave
FS	Adresse »Frequenz« speichern
FU	wie FK, aber korrigiert durch Parameter »Stimmung«
FW	aktueller Frequenz-Wert für SID
H	aktuelle Harmonie-Nummer
H4	1/4-Schlag-Nummer in aktueller Harmonie
HB	Nummer der Harmonie, bis zu der wiederholt werden muß
HD	Dauer der aktuellen Harmonie in 1/4
HL	letzte Harmonie
HM	maximale Anzahl Harmonien
HN	nächste Harmonie
HT	totale Harmonien
HW	Nummer der Harmonie zum Weiterfahren nach Wiederholungen
H$()	Harmonie-Bezeichnungen
HA%()	Akkord-eigene Töne als Bit-Muster
HD%()	Harmonie-Dauer in 1/4 bei Wiederholungen: Wiederholen »bis« Harmonie
HG%()	Nummer Grundton bei Wiederholungen: Wiederholen »von« Harmonie
HS%()	Skala (0 = #-Skala, 1 = b-Skala)
I,J	Laufvariablen
KP	Schulter »keine Percussion«
L$, M0$...MN$	diverse Print-Strings (Striche etc).
MT$	Musik-Titel
N	Nummer des Tones in Bit-Functions
P()	Parameter
PN	Parameter-Nummer
PR$	Print-String während Play-Time
PS$	Print-String für Skala-Striche bei Parametern
R	Adresse im SID-Register
RB	Adresse RUN-Switch »Bass« für IRQ-Programm
RP	Adresse RUN-Switch »Percussion« für ASS-IRQ-Programm
S$()	Ton-Bezeichnung für #- und b-Skala
SM()	Skala-Masken für Bit-Muster generieren
SP()	Skala-Pointer (enthält Grundton-Nummer für A...G)
T0%, T2%, T3%, T4%	Zeiten (in 1/60-Sek) für Vor- und Hauptschläge
TA%	aktive Ton-Nummer
TR%	aktiver Trend (+/-1)
W	Laufvariable für Wiederholungen
WA	Anzahl Wiederholungen
X,Y,Z	Zwischen-Speicher
X$,Z$	Zwischen-Speicher

Tabelle der Variablen in »Bassist«

```
10 REM ***********                          <125>
20 REM * BASSIST *                          <253>
40 REM ***********                          <155>
50 :                                        <026>
60 REM      1985, ROBERT TREICHLER,  FL-94
   97 TRIESENBERG (F.TUM LIECHTENSTEIN)     <176>
70 :                                        <046>
80 POKE 53280,0:POKE 53281,0:PRINT CHR$(14
   )CHR$(8)"{CLR}"                          <213>
90 IF Z=0 THEN Z=1:X$="":Z$="":LOAD"BASS/I
   RQ",8,1:REM ASS-PROG.EINLESEN            <094>
100 :                                       <076>
101 REM DEFINITIONEN & INIT.                <170>
102 :                                       <078>
200 HM=100:DIM H$(HM),HD%(HM),HA%(HM),HS%(
    HM),HG%(HM),F(11),P(4,2)                <078>
205 DIM S$(12,1),SP(6),SM(11),X$(16)        <180>
210 REM FUNC.F.BIT-MUSTER (Y=BIT-MUSTER,Z=
    GRUNDTON,N=NR.1/2-TON UEBER GRUNDTON): <039>
212 DEF FN BS(N)=Y OR SM(N+Z+(N+Z>11)*12):
    REM FUNCT. SET BIT                      <070>
214 DEF FN BC(N)=Y AND 4095-SM(N+Z+(N+Z>11
    )*12):REM FUNCT. CLEAR BIT              <214>
216 DEF FN BT(N)=Y AND SM(N+Z+(N+Z>11)*12)
    :REM FUNCT. TEST BIT                    <202>
220 F=110:FOR I=0 TO 11:F(I)=F:F=F*1.05946
    3094:SM(I)=2↑I:NEXT:REM FREQU.+BIT-MAS
    K.                                      <106>
230 FOR I=0 TO 6:READ SP(I):NEXT:REM SKALA
    -POINTERS (A...G)                       <147>
240 FOR J=0 TO 1:FOR I=0 TO 12:READ S$(I,J
    ):NEXT I,J:REM #- & B-SKALEN            <244>
250 FOR I=0 TO 4:FOR J=0 TO 2:READ P(I,J):
    NEXT J,I:REM PARAMETER                  <136>
260 C$=CHR$(13):R=54272:REM ADR.SID-REG.    <073>
265 MN$="{3SPACE}< MIT 'F1' ZURUECK INS ME
    NUE >"                                  <121>
266 M0$="{7SPACE}@@@@@@@@@@@@@@@@@@@@@@@@@@
    "                                       <077>
267 M1$="{7SPACE}TTTTTTTTTTTTTTTTTTTTTTTTTT
    "                                       <210>
268 M2$="{6SPACE}R "                        <233>
269 M3$="{6SPACE}R{25SPACE}G"               <185>
270 PS$="{2SPACE}-+**+**+**+**+**+**+**+**
    +**+**+"                                <165>
275 L$="{39SPACE}"                          <094>
280 FA=.943874:FK=6.378901:F6=1.019440644:
    FO=1.37:F1=150:F2=300:REM FREQU.KONST   <017>
290 AP=49152:FS=AP+22:RB=251:RP=252:REM AD
    R.F.ASS-PROG                            <130>
300 :                                       <022>
301 REM MENUE                               <235>
302 :                                       <024>
310 GOSUB 6950:Z=32:PRINT"{CLR,CYAN,2DOWN}
    "M0$:PRINT M3$                          <250>
320 PRINT M2$TAB(13)"B A S S I S T"TAB(Z)"
    G":PRINT M3$:PRINT M1$:PRINT M0$:PRINT
     M3$                                    <115>
321 PRINT M2$"1=NEUEINGABE/EDITIEREN"TAB(Z
    )"G":PRINT M3$                          <086>
322 PRINT M2$"2=ABSPEICHERN AUF DISK"TAB(Z
    )"G":PRINT M3$                          <138>
323 PRINT M2$"3=EINLESEN VON DISK"TAB(Z)"G
    ":PRINT M3$                             <107>
324 PRINT M2$"4=PLAY BASS SOLO"TAB(Z)"G":P
    RINT M3$                                <062>
325 PRINT M2$"5=PLAY BASS+SCHLAGZEUG"TAB(Z
    )"G":PRINT M3$                          <010>
326 PRINT M2$"6=PARAMETER-EINSTELLUNG"TAB(
    Z)"G":PRINT M3$                         <099>
327 PRINT M2$"7=ENDE"TAB(Z)"G":PRINT M3$:P
    RINT M1$                                <136>
330 GET Z$:IF Z$<"1"OR Z$>"7"THEN 330       <094>
340 Z=VAL(Z$):ON Z GOSUB 8000,2000,3000,50
    0,505,5000,900                          <229>
350 POKE 198,0:GOTO 300                     <161>
390 END                                     <138>
500 :                                       <222>
501 REM PLAY                                <249>
502 :                                       <224>
504 KP=1:GOTO 510                           <163>
505 KP=0                                    <219>
```

Listing 1. Das Basic-Programm »Bassist« geben Sie bitte mit dem Checksummer ein

```
510 PRINT"{CLR,5SPACE}"MT$:PRINT"{DOWN}"MN
    $"{DOWN}"                               <012>
520 TR%=1:AZ=1:FL=0:H=0:H$(HT+1)=H$(1):PR$
    =""                                     <159>
525 AU=HD%(HT)-A4+1:IF AU<1 THEN AU=1     <050>
530 GOSUB 6900:GOSUB 1800                 <018>
540 FOR W=1 TO WA:HB=-1:HN=1:H4=0         <111>
550 HL=H:H=HN:HN=H+1:IF H>HT THEN 850     <174>
555 IF H<HL THEN PR$="WIEDERHOLUNG"       <207>
560 IF H>HL+1 THEN PR$="ENDE WIEDERHOLUNG" <158>
570 IF HN=HB+1 THEN HN=HW:HB=-1           <070>
580 IF H$(HN)="W"THEN HB=HD%(HN):HW=HN+1:H
    N=HG%(HN)                               <179>
590 FOR HD=1 TO HD%(H):H4=H4+1:IF H4>A4 TH
    EN H4=1                                 <071>
600 :                                     <068>
601 REM TON BESTIMMEN                      <197>
602 :                                     <070>
605 IF H$(H)="PAUSE"THEN 630              <130>
610 IF W<WA OR H<HT OR HD<AU THEN 640     <014>
620 IF HD=AU THEN TA%=HG%(H):GOTO 680:REM
    LETZTER TAKT BASS AUSHALTEN             <161>
630 TA%=12:FW=0:GOTO 720:REM PAUSE        <071>
640 IF(H4 AND 1)OR HD=HD%(H)THEN 650      <210>
645 IF RND(0)<P(3,0)THEN 630              <203>
650 IF HD=1 AND HG%(H)<>HG%(HL)THEN TA%=HG
    %(H):GOTO 680:REM NEUE HARMONIE BEGINN
    T                                       <079>
660 IF HD<>HD%(H)OR HG%(H)=HG%(HN)THEN 670
    :REM ES FOLGT KEINE NEUE HARMONIE       <174>
662 IF H$(HN)="PAUSE"THEN TA%=HG%(H):GOTO
    680:REM PAUSE FOLGT                     <011>
665 SYS AP+12,HA%(H),HA%(HN),HG%(H),HG%(HN
    ),TR%,TA%:GOTO 680:REM UEBERG.TON SUCH
    EN                                      <099>
670 IF RND(0)<P(4,0)THEN SYS AP+15,HA%(H),
    TA%:GOTO 680:REM ZUFALLS-TON            <074>
675 SYS AP+9,HA%(H),TR%,TA%:REM NAE.AKKORD
    EIG.TON SUCHEN                          <253>
680 F=F(TA%):IF F=FL OR F*FO<FL AND F+F<>F
    L THEN F=F+F:REM FREQU.                 <189>
690 TR%=1+2*(F<FL):IF F<F1 THEN TR%=1:REM
    TREND BESTIMMEN                         <055>
695 IF F>F2 THEN TR%=-1:REM ...DABEI ECKFR
    EQU. BEACHTEN                           <112>
700 :                                     <168>
701 REM TON AUSGEBEN                       <125>
702 :                                     <170>
710 FL=F:FW=F*FU:REM FREQU.IN SID-WERT UMR
    ECHNEN                                  <254>
720 IF(HD>1 OR H$(H)=H$(HL))AND H4>1 THEN
    800                                     <199>
730 IF AZ THEN GOSUB 1500:AZ=0            <243>
740 IF PEEK(FS)THEN 740:REM WARTE BIS LETZ
    TER TON V.IRQ-ROUT.BEHANDELT ...        <025>
750 IF PR$>""THEN PRINT:PRINT:PRINT PR$:PR
    $=""                                    <106>
760 PRINT:PRINT H$(H)TAB(9)": ";:REM ..ERS
    T DANN HARMONIE-BEZ. AUSGEBEN           <164>
800 SYS AP+6,H4,FW,FW*FA,S$(TA%,HS%(H)):RE
    M TON MIT BEZ. ->ASS-PROG               <113>
820 IF PEEK(197)=4 THEN HD=HD%(H):HN=HT+1:
    W=WA:REM ABBRUCH                        <198>
830 NEXT HD:GOTO 550                      <172>
850 NEXT W:SYS AP+6,0,0,0,"":SYS AP+3:REM
    IRQ-ROUT. AUS                           <190>
890 FOR I=0 TO 1500:NEXT:RETURN           <250>
900 :                                     <114>
901 REM ENDE                               <176>
902 :                                     <116>
990 END                                    <230>
1300 :                                    <006>
1301 REM WERTE ZU EINER HARMONIE GENERIERE
     N                                      <240>
1302 :                                    <008>
1308 Z=SP(ASC(X$)-193):Z$=MID$(X$,2,1):X=0 <210>
1310 IF Z$="#"THEN Z=Z+1:GOTO 1335        <078>
1320 IF Z$="B"THEN Z=Z-1-(Z<1)*12:X=1:GOTO
     1335                                   <111>
1330 IF Z$<>" "THEN X$=LEFT$(X$,1)+" "+MID
     $(X$,2)                                <032>
1332 IF Z=3 OR Z=8 THEN X=1:REM C- & F-SKA
     LEN MIT B (NICHT #)                    <194>
1335 H$(HT)=X$:HG%(HT)=Z:HS%(HT)=X:REM BEZ
     ./GRUNDTON/SKALA (# ODER B)            <020>
1340 FOR I=0 TO HT-1:IF H$(I)=X$THEN Y=HA%
     (I):GOTO 1400                          <194>
1345 NEXT I:Y=0                           <049>
1350 Y=FN BS(0)+FN BS(4)+FN BS(7):REM BIT-
     MUSTER F.GRUND-DREIKLANG               <180>
1355 FOR I=2 TO LEN(X$):Z$=MID$(X$,I,1)   <116>
1360 IF Z$="M"THEN Y=FN BC(4):Y=FN BS(3):G
     OTO 1399:REM MOLL                      <111>
1365 IF Z$="J"THEN Y=FN BC(10):Y=FN BS(11)
     :GOTO 1399:REM MAJOR                   <054>
1370 IF Z$="+"THEN Y=FN BC(7):Y=FN BS(8):G
     OTO 1399:REM QUINTE +                  <219>
1375 IF Z$="-"THEN Y=FN BC(7):Y=FN BS(6):G
     OTO 1399:REM QUINTE -                  <225>
1380 IF Z$="0"OR Z$="O"THEN Y=FN BC(4):Y=F
     N BS(3):Y=FN BC(7):Y=FN BS(6):REM VER
     MINDERT                                <170>
1382 IF Z$="6"THEN Y=FN BS(9):GOTO 1399:RE
     M SEXTE                                <242>
1385 IF Z$="7"THEN 1398                   <150>
1386 IF Z$="9"THEN 1397                   <119>
1387 IF Z$="1"THEN 1396                   <084>
1388 IF Z$="3"THEN 1395                   <054>
1390 GOTO 1399                            <207>
1395 Y=FN BS(9):REM 13-ER                 <168>
1396 Y=FN BS(5):REM 11-ER                 <101>
1397 Y=FN BS(2):REM  9-ER                 <157>
1398 IF FN BT(11)=0 THEN Y=FN BS(10):REM 7
     -ER, WENN NICHT SCHON MAJOR-7          <047>
1399 NEXT I                               <213>
1400 HA%(HT)=Y:PRINT:PRINT X$TAB(9)": ";  <156>
1410 FOR I=0 TO 11:IF FN BT(I)=0 THEN 1450 <179>
1420 PRINT S$(I+Z+(I+Z>11)*12,X)" ";      <120>
1450 NEXT I:PRINT:RETURN                  <179>
1500 :                                    <206>
1501 REM ANZAEHLEN                         <183>
1502 :                                    <208>
1510 SYS AP:REM INIT.IRQ-PROG.            <254>
1511 PRINT:PRINT"STIMMTON : "S$(TA%,HS%(H)
     )                                      <051>
1512 POKE RB,1:Z=FW:FOR I=1 TO A4         <036>
1514 SYS AP+6,128,Z,0,"":Z=0:NEXT         <183>
1515 IF PEEK(FS)THEN 1515                 <164>
1516 PRINT:PRINT"ANZAEHLEN:";             <136>
1520 POKE RP,1:FOR I=1 TO A4              <194>
1530 SYS AP+6,128,0,0,STR$(I):NEXT        <122>
1540 IF KP THEN POKE RP,0                 <007>
1550 PR$=" ":RETURN                       <117>
1800 :                                    <254>
1801 REM FREQU.UMRECHN.KONST. & TEMPO RECH
     NEN/->ASS.PROG.                        <213>
1802 :                                    <000>
1820 FU=FK*F6↑P(0,0):T0%=0                <196>
1840 REM VORSCHLAEGE BEI TEMPI<130 ->1/16-
     NOTEN, WENN SCHNELLER ->1/8-TRIOLEN    <183>
1850 IF P(1,0)<130 THEN Z=INT(900/P(1,0)+.
     5):T2%=2*Z:T3%=T2%+Z:T4%=T3%+Z:GOTO 1
     870                                    <121>
1860 Z=INT(1200/P(1,0)+.5):T2%=Z:T3%=T2%+Z
     :T4%=T3%+Z:GOTO 1870                   <176>
1870 POKE R+5,10+T4%/50:REM BASS-DECAY AUF
     GRUND TEMPO                            <052>
1880 SYS AP+18,T2%,T3%,T4%:RETURN         <092>
1897 :                                    <095>
1898 REM TON -> SID                        <117>
1899 :                                    <097>
2000 :                                    <198>
2001 REM ABSPEICHERN AUF DISK              <174>
2002 :                                    <200>
2010 PRINT"{CLR,DOWN}ABSPEICHERN AUF DISK" <187>
2020 INPUT"{2DOWN}MUSIK-TITEL";MT$        <192>
2030 GOSUB 2700:IF ER THEN RETURN         <232>
2040 OPEN 2,8,2,MT$+",S,W":GOSUB 2800:IF E
     R=0 THEN 2050                          <144>
2042 IF ER<>63 THEN RETURN                <062>
2044 PRINT"{DOWN}UEBERSCHREIBEN (J/N)? "; <054>
2046 GET Z$:IF Z$<>"J"AND Z$<>"N"THEN 2046 <090>
2048 PRINT Z$:IF Z$="N"THEN 2090          <182>
2049 CLOSE 2:PRINT#15,"S0:"+MT$:GOTO 2040 <083>
2050 PRINT#2,HT;C$;A4;C$;WA:FOR I=0 TO 4:P
     RINT#2,P(I,0):NEXT                     <120>
2060 GOSUB 2800:IF ER THEN RETURN         <024>
2070 FOR I=1 TO HT:PRINT#2,H$(I);C$;HA%(I)
     ;C$;HS%(I);C$;HG%(I);C$;HD%(I):NEXT I  <050>
2090 GOSUB 2800:CLOSE 2:CLOSE 15:RETURN   <050>
```

Listing 1. »Bassist« (Fortsetzung)

```
2600 :                                          <036>
2700 OPEN 15,8,15,"I0"                          <033>
2800 INPUT#15,ER,ER$,ET,ES:IF ER=0 THEN RE
     TURN                                       <251>
2820 PRINT:PRINT ER;ER$;ET;ES:IF ER<20 OR
     ER=63 THEN RETURN                          <210>
2850 :                                          <032>
2900 CLOSE 2:CLOSE 15                           <026>
2910 :                                          <092>
2950 PRINT:PRINT"{2DOWN}<TASTE DRUECKEN>":
     POKE 198,0                                 <052>
2960 GET Z$:IF Z$=""THEN 2960                   <017>
2980 RETURN                                     <244>
3000 :                                          <182>
3001 REM EINLESEN VON DISK                      <108>
3002 :                                          <184>
3010 PRINT"{CLR}TITEL EINGEBEN ODER 'RETUR
     N' FUER":PRINT"{DOWN}INHALTSVERZEICHN
     IS{DOWN}"                                  <071>
3020 MT$="":INPUT MT$:IF MT$>""THEN 3040        <024>
3030 GOSUB 3500:IF ER OR MT$=""THEN RETURN      <088>
3040 GOSUB 2700:OPEN 2,8,2,"0:"+MT$+",S,R"
     :GOSUB 2800:IF ER THEN RETURN              <078>
3050 INPUT#2,HT,A4,WA:FOR I=0 TO 4:INPUT#2
     ,P(I,0):IF P(I,0)<P(I,1)THEN P(I,0)=P
     (I,1)                                      <136>
3055 IF P(I,0)>P(I,2)THEN P(I,0)=P(I,2)         <212>
3060 NEXT I:GOSUB 2800:IF ER THEN RETURN        <252>
3065 PRINT"{CLR}"MT$"{2SPACE}:"STR$(A4)"/4
     -TAKT":PRINT"{DOWN}HARMONIE{2SPACE}BA
     UER IN 1/4{DOWN}"                          <135>
3070 FOR I=1 TO HT:INPUT#2,H$(I):Z=ASC(H$(
     I)+CHR$(0)):IF Z=87 OR Z=208 THEN 308
     0:REM W+P                                  <052>
3075 IF Z<193 OR Z>199 THEN PRINT:PRINT"**
     * FILE-FEHLER ***":GOTO 2900               <181>
3080 INPUT#2,HA%(I),HS%(I),HG%(I),HD%(I)        <206>
3090 PRINT H$(I)TAB(12)HD%(I):NEXT I:FOR I
     =0 TO 500:NEXT:GOTO 3900                   <004>
3500 :                                          <174>
3510 GOSUB 2700:IF ER THEN RETURN               <188>
3515 OPEN 2,8,2,"#":DT=18:DS=1                  <249>
3520 PRINT#15,"U1";2;0;DT;DS                    <223>
3522 PRINT"{CLR}INHALT:":PRINT"{DOWN}NR TI
     TEL":PRINT"TT TTTTT{DOWN}"                 <031>
3525 PRINT#15,"B-P";2;0:GET#2,Z$:DT=ASC(Z$
     +CHR$(0)):GET#2,Z$:DS=ASC(Z$+CHR$(0))      <022>
3530 FOR I=0 TO 7:PRINT#15,"B-P";2;I*32+2       <045>
3540 GET#2,Z$:IF Z$<>CHR$(129)THEN 3600         <165>
3550 GET#2,Z$,Z$                                <087>
3560 X$="":FOR J=1 TO 16:GET#2,Z$:X$=X$+Z$
     :NEXT:Z$(I)=X$:PRINT I;X$                  <120>
3600 NEXT:PRINT"{3DOWN}TASTE DRUECKEN:{DOW
     N}":PRINT"NR.0...7 = DIESEN TITEL EIN
     LESEN"                                     <244>
3610 PRINT"'SPACE'{2SPACE}= WEITER IM INHA
     LTSVERZEICHNIS"                            <120>
3620 PRINT"'F1'{5SPACE}= ZURUECK INS MENUE
     "                                          <240>
3640 GET Z$:IF Z$=""THEN 3640                   <029>
3650 IF Z$=CHR$(133)THEN MT$="":GOTO 3900       <212>
3660 IF Z$=>"0"AND Z$<"8"THEN MT$=Z$(VAL(Z
     $)):GOTO 3900                              <175>
3670 IF Z$<>" "THEN 3640                        <222>
3680 IF DT<1 OR DT>35 THEN PRINT"{DOWN}KEI
     NE WEITEREN TITEL":GOTO 3640               <058>
3690 GOTO 3520                                  <234>
3900 CLOSE 2:CLOSE 15:RETURN                    <097>
5000 :                                          <150>
5001 REM PARAMETER AENDERN                      <234>
5002 :                                          <152>
5020 PRINT"{CLR}PARAMETER-EINSTELLUNGEN:"       <026>
5030 PRINT"NR. WAEHLEN (1-5) UND MIT +/- A
     ENDERN"                                    <119>
5100 PRINT"{DOWN}1) {10SPACE,RVSON}STIMMUNG
     {RVOFF,SPACE}(A=220 HZ)"                   <174>
5110 PRINT"{3SPACE}E{2SPACE}F{2SPACE}F# G{
     2SPACE}G# A{2SPACE}A# B{2SPACE}C{2SPA
     CE}C# D":PRINT PS$                         <071>
5120 PRINT"{2DOWN}2) 30 45 60{SPACE,RVSON}
     TEMPO{RVOFF,3SPACE}120{3SPACE}150{3SP
     ACE}180":PRINT PS$                         <155>
5130 PRINT"{2DOWN}3) DUNKEL{3SPACE,RVSON}B
     ASS-FILTER{RVOFF,7SPACE}HELL":PRINT P
     S$                                         <238>
5140 PRINT"{2DOWN}4) 1/4{6SPACE,RVSON}BASS
     -NOTEN{RVOFF,9SPACE}1/2":PRINT PS$         <190>
5150 PRINT"{2DOWN}5) TREND{4SPACE,RVSON}BA
     SS-LINIE{RVOFF,6SPACE}ZUFALL":PRINT P
     S$                                         <035>
5160 PRINT"{2DOWN}"MN$"{HOME}"                  <000>
5165 :                                          <061>
5170 FOR PN=0 TO 4:GOSUB 5500:NEXT              <005>
5180 GOSUB 6900:PN=0:F=220                      <006>
5190 GOSUB 1800:SYS AP:SYS AP+6,0,F*FU,0,"
     ":POKE RB,1                                <144>
5200 Z=PEEK(197):IF Z=4 THEN 5900               <031>
5210 IF Z=56 THEN PN=0                          <234>
5220 IF Z=59 THEN PN=1                          <189>
5230 IF Z=8 THEN PN=2                           <084>
5240 IF Z=11 THEN PN=3                          <095>
5250 IF Z=16 THEN PN=4                          <178>
5290 IF Z<>40 AND Z<>43 THEN 5200               <164>
5300 I=(P(PN,2)-P(PN,1))/30:IF Z=43 THEN I
     =-I                                        <146>
5310 P(PN,0)=P(PN,0)+I                          <253>
5320 IF P(PN,0)<P(PN,1)THEN P(PN,0)=P(PN,1
     )                                          <058>
5330 IF P(PN,0)>P(PN,2)THEN P(PN,0)=P(PN,2
     )                                          <062>
5400 :                                          <042>
5410 GOSUB 5500:IF PN=0 OR PN=1 THEN 5190       <155>
5415 IF PN=2 THEN POKE R+22,P(2,0)              <115>
5420 GOTO 5200                                  <090>
5490 :                                          <132>
5500 PRINT"{HOME,DOWN}":FOR I=0 TO PN:PRIN
     T"{4DOWN}";:NEXT                           <183>
5510 Z=INT(30/(P(PN,2)-P(PN,1))*(P(PN,0)-P
     (PN,1))+3.5)                               <052>
5520 PRINT LEFT$(L$,Z)"↑"LEFT$(L$,35-Z)         <097>
5530 RETURN                                     <254>
5900 :                                          <034>
5910 SYS AP+3:RETURN                            <153>
6900 :                                          <016>
6901 REM INIT.SID-REGISTERS: BASS=VCO#1+2,
     PERC=VCO#3                                 <039>
6902 :                                          <018>
6910 GOSUB 6950:POKE R+10,1:POKE R+12,8:RE
     M VCO#2                                    <085>
6920 POKE R+15,80:REM VCO#3                     <193>
6930 POKE R+22,P(2,0):POKE R+23,240+8+2+1:
     POKE R+24,16+15:REM FILTER & VOLUME        <090>
6940 RETURN                                     <140>
6950 FOR I=R+24 TO R STEP-1:POKE I,0:NEXT:
     RETURN                                     <116>
8000 :                                          <102>
8001 REM NEUEINGABE / EDITIEREN                 <097>
8002 :                                          <104>
8100 IF HT<1 THEN PRINT"{CLR}NEUEINGABE":G
     OTO 8150                                   <140>
8102 PRINT"{CLR}NEUEINGABE ODER EDITIEREN
     (N/E)?";                                   <161>
8105 GET Z$:IF Z$=""THEN 8105                   <019>
8110 PRINT Z$:IF Z$="N"THEN 8150                <144>
8120 IF Z$="E"THEN HL=HT:GOTO 8200              <087>
8130 GOTO 8100                                  <012>
8150 HL=0:A4=4:WA=1                             <017>
8200 HT=0:H=0                                   <099>
8210 PRINT"{DOWN}ANZAHL 1/4 PRO TAKT{2SPAC
     E}"A4:PRINT TAB(20)"{UP}";:INPUT A4        <052>
8220 PRINT"{DOWN}ANZAHL GESAMT-WIEDERHOLUN
     GEN{2SPACE}"WA:PRINT TAB(29)"{UP}";:I
     NPUT WA                                    <023>
8300 PRINT"{CLR}HARMONIEFOLGE EINGEBEN (*=
     ENDE EINGABEN){DOWN}"                      <026>
8305 PRINT" P=PAUSE{12SPACE}W=WIEDERHOLUNG
     {DOWN}"                                    <078>
8310 PRINT" A...H (ODER A...H){2SPACE}=GRU
     ND-DREIKLANG"                              <021>
8315 PRINT" #=ERHOEHT{10SPACE}B=ERNIEDRIGT
     "                                          <229>
8320 PRINT" J=MAJOR{12SPACE}M=MOLL"             <167>
8330 PRINT" +=ERHOEHTE QUINT{3SPACE}-=ERNI
     EDRIGTE QUINT"                             <215>
8340 PRINT" O=VERMINDERT{7SPACE}B=DEUTSCHE
     S H"                                       <116>
8350 PRINT" 6,7,9,11,13=SEXT,SEPT,NONE,USW
     ."                                         <017>
```

Listing 1. »Bassist« (Fortsetzung)

```
8360 PRINT"{DOWN}BEISPIELE:"                   <054>
8365 PRINT" C{4SPACE}=C-DUR-AKKORD"            <218>
8370 PRINT" F#M9 =FIS-MOLL-NONE-AKKORD"        <115>
8380 PRINT" GB6/7=GES-SEXT/SEPT-AKKORD"        <102>
8390 PRINT" BB7J =BES-MAJOR-SEPT-AKKORD{DO
     WN}"                                      <133>
8400 X$="":HD=4:IF HT<HL THEN X$=H$(HT+1):
     HD=HD%(HT+1)                              <041>
8410 PRINT"{2DOWN}TAKT"RIGHT$("{2SPACE}"+S
     TR$(INT(H/A4)+1),3)":{3SPACE}"X$:PRIN
     T TAB(9)"{UP}";                           <029>
8420 INPUT X$:X=ASC(X$+CHR$(0))AND 127         <092>
8430 IF X=42 THEN 8900:REM * (ENDE)            <208>
8435 IF X=87 THEN 8500:REM W (WIEDERH.)        <134>
8440 IF X=72 THEN X=66:REM H WIRD B            <247>
8445 IF X=80 THEN HT=HT+1:H$(HT)="PAUSE":G
     OTO 8480:REM P (PAUSE)                    <083>
8450 IF X<65 OR X>71 THEN PRINT"{DOWN}***
     FEHLER ***":GOTO 8400                     <255>
8460 X$=CHR$(X+128)+MID$(X$,2)                 <222>
8470 HT=HT+1:GOSUB 1300                        <098>
8480 PRINT"{DOWN}DAUER IN 1/4{2SPACE}"HD:P
     RINT TAB(13)"{UP}";:INPUT HD::H=H+HD      <099>
8490 HD%(HT)=HD:GOTO 8400                      <159>
8500 :                                         <094>
8501 REM WIEDERHOLUNGSZEICHEN BEARBEITEN       <030>
8510 IF HT=>HL THEN HD=HT:HG%(HT+1)=1          <172>
8520 HT=HT+1:H$(HT)="W"                        <106>
8530 PRINT"{DOWN}AB DER WIEVIELTEN HARMONI
     E"                                        <093>
8535 PRINT"SOLL WIEDERHOLT WERDEN"             <160>
8537 PRINT" "HG%(HT):PRINT"{UP}";:INPUT Z:
     IF Z>HT-2 OR Z<1 THEN 8530                <152>
8538 HG%(HT)=Z                                 <129>
8540 PRINT"{DOWN}BIS (UND MIT) ZUR WIEVIEL
     TEN HARMONIE"                             <140>
8545 PRINT"SOLL WIEDERHOLT WERDEN ( LETZTE
     ="HT-1")"                                 <047>
8550 PRINT" "HD:PRINT"{UP}";:INPUT Z:IF Z=
     >HT OR Z<=HG%(HT)THEN 8530                <241>
8560 HD%(HT)=Z:GOTO 8400                       <068>
8900 :                                         <240>
8910 IF HT<HL THEN HT=HL:REM EXIT, KORR.HT
      WENN EDIT.                               <149>
8920 RETURN                                    <086>
9000 :                                         <086>
9100 :                                         <186>
9101 REM SKALA-POINTERS A,B,C,D,E,F,G          <166>
9102 :                                         <188>
9110 DATA 0,2,3,5,7,8,10                       <178>
9200 :                                         <030>
9201 REM # UND B-SKALEN CHROMAT.               <068>
9202 :                                         <032>
9210 DATA"A ","A#","B ","C ","C#","D ","D#
     ","E ","F ","F#","G ","G#","£ "           <124>
9220 DATA"A ","BB","B ","C ","DB","D ","EB
     ","E ","F ","GB","G ","AB","£ "           <253>
9400 :                                         <232>
9401 REM PARAM. STD/TIEFST/HOECHST-WERTE       <169>
9402 :                                         <234>
9410 DATA 15,0,30:REM STIMMUNG (1/6-TOENE
     OBERHALB 'E' ->'E'...'D')                 <108>
9420 DATA 105,30,180:REM TEMPO                 <246>
9430 DATA 15,0,30:REM BASS-FILTER              <048>
9440 DATA.2,0,1:REM 1/4 - 1/2 NOTEN            <254>
9450 DATA.2,0,1:REM TREND - ZUFALL             <154>
9480 :                                         <058>
50000 :                                        <191>
50001 REM SAVE PROG.                           <133>
50002 :                                        <193>
50010 OPEN 15,8,15,"S0:BASSIST":GOSUB 2800     <097>
50020 SAVE"BASSIST",8:GOSUB 2800:CLOSE 15      <005>
50030 END                                      <247>
```

© 64'er

Listing 1. »Bassist« (Schluß)

64ER ONLINE

```
programm : bass/irq          c000 c2c3
-------------------------------------------

c000 : 4c 37 c0 4c 4d c0 4c 57   5c
c008 : c1 4c b1 c1 4c d2 c1 4c   8f
c010 : 88 c2 4c b0 c2 00 00 00   4f
c018 : 00 00 00 00 00 00 00 00   19
c020 : 00 00 00 00 00 00 00 00   21
c028 : 00 00 00 01 02 04 08 10   c9
c030 : 20 40 80 01 02 04 08 a9   64
c038 : 54 a2 c0 78 8d 14 03 8e   bf
c040 : 15 03 a9 00 85 fb 85 fc   89
c048 : 8d 16 c0 58 60 a9 31 a2   79
c050 : ea 4c 3b c0 a5 fb 05 fc   8f
c058 : f0 15 ee 22 c0 ad 22 c0   56
c060 : cd 1f c0 f0 10 cd 20 c0   7c
c068 : f0 2d cd 21 c0 f0 57 8d   92
c070 : 22 c0 4c 31 ea a9 00 8d   43
c078 : 23 c0 ad 15 c0 30 f3 f0   49
c080 : f1 29 01 d0 ed ad 12 d0   96
c088 : 69 dc b0 e6 a9 05 8d 23   a7
c090 : c0 20 0d c1 4c 72 c0 ad   93
c098 : 15 c0 30 d6 f0 d4 29 01   51
c0a0 : f0 05 a9 05 8d 23 c0 20   53
c0a8 : 0d c1 ad 15 c0 c9 02 b0   68
c0b0 : c1 ad 12 d0 69 c8 b0 ba   fc
c0b8 : ad 19 c0 ac 18 c0 f0 b2   68
c0c0 : 20 27 c1 4c 72 c0 a2 05   2f
c0c8 : ad 15 c0 29 01 f0 02 a2   3a
c0d0 : 08 8e 23 c0 20 0d c1 ad   cd
c0d8 : 17 c0 ac 16 c0 20 27 c1   6a
c0e0 : a9 00 8d 22 c0 8d 23 c0   b7
c0e8 : ae 15 c0 f0 1d 8d 16 c0   87
c0f0 : 8d 18 c0 ad c3 c2 f0 12   a9
c0f8 : a2 00 bd c3 c2 f0 06 20   8e
c100 : d2 ff e8 d0 f5 a9 20 20   93
c108 : d2 ff 4c 72 c0 a5 fc f0   4a
c110 : 15 ad 23 c0 f0 10 a9 80   14
c118 : 8d 12 d4 ad 23 c0 8d 13   2e
c120 : d4 a9 81 8d 12 d4 60 d0   c6
c128 : 07 a9 2a 20 d2 ff a9 ff   66
c130 : 30 24 a6 fb f0 20 a2 20   76
c138 : 8e 04 d4 a2 40 8e 0b d4   a0
c140 : 8d 00 d4 8c 01 d4 8d 07   8f
c148 : d4 8c 08 d4 a9 21 8d 04   e1
c150 : d4 a9 43 8d 0b d4 60 20   94
c158 : fd ae 20 9e b7 8e 1a c0   62
c160 : 20 b2 b1 a5 64 d0 02 a9   22
c168 : ff 8d 1b c0 a5 65 8d 1c   01
c170 : c0 20 b2 b1 a5 64 8d 1d   11
c178 : c0 a5 65 8d 1e c0 ad 16   e1
c180 : c0 f0 05 ad 15 c0 d0 f6   38
c188 : a2 04 bd 1a c0 9d 15 c0   ae
c190 : ca 10 f7 20 fd ae 20 9e   77
c198 : ad 20 a3 b6 aa a0 00 e8   96
c1a0 : ca f0 08 b1 22 99 c3 c2   9e
c1a8 : c8 d0 f5 a9 00 99 c3 c2   ec
c1b0 : 60 20 3f c2 8d 24 c0 8e   63
c1b8 : 25 c0 20 3f c2 8d 29 c0   ec
c1c0 : 20 3f c2 8d 2a c0 ad 29   94
c1c8 : c0 20 65 c2 f0 f8 20 59   54
c1d0 : c2 60 20 3f c2 8d 24 c0   5d
c1d8 : 8e 25 c0 20 3f c2 8d 26   ba
c1e0 : c0 8e 27 c0 20 3f c2 8d   eb
c1e8 : 28 c0 20 3f c2 8d 2a c0   23
c1f0 : 20 3f c2 8d 29 c0 20 3f   aa
c1f8 : c2 8d 1a c0 a9 ff 20 65   05
c200 : c2 d0 32 a9 02 20 65 c2   28
c208 : d0 2b a9 fd 20 65 c2 d0   72
c210 : 24 a9 04 20 65 c2 d0 1d   f8
c218 : ad 24 c0 2d 26 c0 8d 24   94
c220 : c0 ad 25 c0 2d 27 c0 8d   42
c228 : 25 c0 0d 24 c0 d0 0a ad   8b
c230 : 28 c0 8d 2a c0 20 59 c2   59
c238 : 60 ad 1a c0 4c c3 c1 20   38
c240 : fd ae 20 8b b0 85 49 84   73
c248 : 4a a5 0e f0 09 a0 00 b1   ff
c250 : 49 aa c8 b1 49 60 4c 99   53
c258 : ad a9 00 a8 91 49 ad 2a   5d
c260 : c0 c8 91 49 60 18 6d 2a   e3
c268 : c0 10 03 18 69 0c c9 0c   2a
c270 : 90 03 38 e9 0c 8d 2a c0   24
c278 : aa bd 2b c0 a0 00 e0 08   81
c280 : 90 02 a0 01 39 24 c0 60   d2
c288 : 20 3f c2 8d 24 c0 8e 25   77
c290 : c0 20 3f c2 20 75 c2 49   d4
c298 : ff 39 24 c0 99 24 c0 ad   6e
c2a0 : 12 d0 29 07 d0 02 a9 01   0b
c2a8 : 20 65 c2 f0 f9 4c 59 c2   36
c2b0 : 20 3f c2 8d 1f c0 20 3f   c9
c2b8 : c2 8d 20 c0 20 3f c2 8d   83
c2c0 : 21 c0 60 00 ff ff ff 00   59
```

Listing 2. Das Maschinen-Programm »BASS/IRQ« steuert den SID. Bitte mit dem MSE eingeben.

```
                ;
                ;*************
                ;* bass/irq *
                ;*************
                ;
                ; 1985 robert treichler
                ;          fl-9497 triesenberg, f.tum liechtenstein
                ;
220:   c000            *=   $c000
                ;
                ; aufrufe aus basic ---------------
                ;
                ; init     sys ap
                ; exit     sys ap+3
                ; para     sys ap+6,h4,fw,fw*fw,ton-bez.
                ; trend    sys ap+9,ha%(h),tr%,ta%
                ; hnext    sys ap+12,ha%(h),ha%(hn),hg%(h),hg%(hn),tr%,ta%
                ; zufall   sys ap+15,ha%(h),ta%
                ; tempo    sys ap+18,t2%.t3%,t4%
                ;
```

Listing 3. Das Quellcode-Listing zu »BASS/IRQ«

```
c000 4c 37 c0             jmp  init    ;irq-rout. ein
c003 4c 4d c0             jmp  exit    ;irq-rout. aus
c006 4c 57 c1             jmp  para    ;ton-parameter aus basic holen
c009 4c b1 c1             jmp  trend   ;nae.akkordeig.ton suchen
c00c 4c d2 c1             jmp  hnext   ;ueberg.ton zu nae.harm.suchen
c00f 4c 88 c2             jmp  zufall  ;zufalls-ton ermitteln
c012 4c b0 c2             jmp  tempo   ;tempo aus basic holen
                ;
                ; definitionen --------------------
                ;
c015 00         h4        .byt 0       ;nr. 1/4-schlag im takt
c016 00 00      fs        .byt 0,0     ;frequenz hauptschlag
c018 00 00      fv        .byt 0,0     ;frequenz vorschlag
c01a 00 00 00 save        .byt 0,0,0,0,0 ;save h4,fs,fv
                ;
c01f 00         t2        .byt 0       ;zeit-inkrement (1.vors.)
c020 00         t3        .byt 0       ;      do.      (2.vors.)
c021 00         t4        .byt 0       ;      do.      (haupts.)
                ;
c022 00         timer     .byt 0       ;zeit-zaehler
                ;
c023 00         pc        .byt 0       ;perc. attack/decay
                ;
c024 00 00      ha        .byt 0,0     ;akkordeig.toene akt.harmonie (lb/hb)
c026 00 00      hanx      .byt 0,0     ;akkordeig.toene naechste harmonie
c028 00         hg        .byt 0       ;nr.grundton akt.harmonie
c029 00         tr        .byt 0       ;trend +/-1 (1,255)
c02a 00         ta        .byt 0       ;nr.akt.ton
                ;
                ;and-masken fuer 2er-potenzen
c02b 01 02 04 mask        .byt 1,2,4,8,16,32,64,128  ;lb(bit0-7)
c033 01 02 04             .byt 1,2,4,8  ;hb(bit8-11)
                ;
c037            rb        =    251     ;run bass
c037            rp        =    252     ;run percussion
                ;
c037            sid       =    54272   ;sid-reg.adr
c037            random    =    $d012   ;pseudo-random-wert
c037            irqex     =    $ea31   ;irq-rout.exit
c037            chkcom    =    $aefd   ;check komma
c037            chrout    =    $ffd2   ;char-output
c037            getbyt    =    $b79e   ;holt 1-byte-wert ->reg.x
c037            getvar    =    $b08b   ;variable suchen
c037            typerr    =    $ad99   ;type-mismatch-error
c037            getpar    =    $b1b2   ;holt 16-bit-parameter ->$64/65
c037            frmevl    =    $ad9e   ;bel.ausdruck auswerten
c037            frestr    =    $b6a3   ;string-verwaltung
                ;
                ; programm -----------------------
                ;
                ; irq-routine einschalten
                ;
c037 a9 54      init      lda  #<irq
c039 a2 c0                ldx  #>irq
c03b 78         vektor    sei
c03c 8d 14 03             sta  $0314
c03f 8e 15 03             stx  $0315
c042 a9 00                lda  #0
c044 85 fb                sta  rb
c046 85 fc                sta  rp
c048 8d 16 c0             sta  fs
c04b 58                   cli
c04c 60                   rts
                ;
                ;irq-rout. aus
                ;
c04d a9 31      exit      lda  #<irqex
c04f a2 ea                ldx  #>irqex
c051 4c 3b c0             jmp  vektor
                ;
                ;irq-einsprung
                ;
c054 a5 fb      irq       lda  rb
c056 05 fc                ora  rp
c058 f0 15                beq  tim
c05a ee 22 c0             inc  timer
c05d ad 22 c0             lda  timer
c060 cd 1f c0             cmp  t2      ;check intervall-zeiten
c063 f0 10                beq  playt2
c065 cd 20 c0             cmp  t3
c068 f0 2d                beq  playt3
c06a cd 21 c0             cmp  t4
c06d f0 57                beq  playt4
c06f 8d 22 c0 tim         sta  timer
c072 4c 31 ea return      jmp  irqex
                ;
c075 a9 00      playt2    lda  #0      ;1.vorschlag
c077 8d 23 c0             sta  pc
c07a ad 15 c0             lda  h4      ;kein 1.vorschlag, wenn ...
c07d 30 f3                bmi  return  ;...h4=neg.
c07f f0 f1                beq  return  ;...oder h4=0
c081 29 01                and  #1
c083 d0 ed                bne  return  ;...oder schlag=ungerade
c085 ad 12 d0             lda  random
c088 69 dc                adc  #220
c08a b0 e6                bcs  return  ;...oder random-exit
c08c a9 05                lda  #5
c08e 8d 23 c0             sta  pc      ;hi-hat kurz
c091 20 0d c1             jsr  perc
c094 4c 72 c0             jmp  return
                ;
c097 ad 15 c0 playt3      lda  h4      ;2.vorschlag
c09a 30 d6                bmi  return  ;kein 2.vors.wenn h4=neg
c09c f0 d4                beq  return  ;...oder h4=0
c09e 29 01                and  #1
c0a0 f0 05                beq  p310
c0a2 a9 05                lda  #5      ;hi-hat kurz,wenn...
c0a4 8d 23 c0             sta  pc      ;...schlag=ungerade
c0a7 20 0d c1 p310        jsr  perc    ;...oder 1.vors.ausgefuehrt
c0aa ad 15 c0             lda  h4
c0ad c9 02                cmp  #2
c0af b0 c1                bcs  return  ;bass-vorschlag nur bei #1
c0b1 ad 12 d0             lda  random
c0b4 69 c8                adc  #200
c0b6 b0 ba                bcs  return  ;random-exit
c0b8 ad 19 c0             lda  fv+1    ;bass-vorschlag
c0bb ac 18 c0             ldy  fv
c0be f0 b2                beq  return  ;ton noch nicht bereit
c0c0 20 27 c1             jsr  bass
c0c3 4c 72 c0             jmp  return
                ;
c0c6 a2 05      playt4    ldx  #5      ;1/4-hauptschlag
c0c8 ad 15 c0             lda  h4
c0cb 29 01                and  #1
c0cd f0 02                beq  p410
c0cf a2 08                ldx  #8
c0d1 8e 23 c0 p410        stx  pc
c0d4 20 0d c1             jsr  perc
c0d7 ad 17 c0             lda  fs+1    ;bass-hauptschlag
c0da ac 16 c0             ldy  fs
c0dd 20 27 c1             jsr  bass
c0e0 a9 00                lda  #0
c0e2 8d 22 c0             sta  timer   ;reset timer
c0e5 8d 23 c0             sta  pc      ;reset perc.byte
c0e8 ae 15 c0             ldx  h4
c0eb f0 1d                beq  p600
c0ed 8d 16 c0             sta  fs      ;freigeben freq-loc. wenn h4>0
c0f0 8d 18 c0             sta  fv
c0f3 ad c3 c2             lda  string
c0f6 f0 12                beq  p600
c0f8 a2 00                ldx  #0      ;string ausdrucken
c0fa bd c3 c2 p500        lda  string,x
c0fd f0 06                beq  p550
c0ff 20 d2 ff             jsr  chrout
c102 e8                   inx
c103 d0 f5                bne  p500
c105 a9 20      p550      lda  #32
c107 20 d2 ff             jsr  chrout
c10a 4c 72 c0 p600        jmp  return
                ;
c10d a5 fc      perc      lda  rp      ;evtl.percussion ->sid
c10f f0 15                beq  percex  ;->keine perc.
c111 ad 23 c0             lda  pc
c114 f0 10                beq  percex  ;->keine perc.
c116 a9 80                lda  #128
c118 8d 12 d4             sta  sid+18  ;vco#3 noise+gate
c11b ad 23 c0             lda  pc
c11e 8d 13 d4             sta  sid+19  ;vco#3 attack/decay
c121 a9 81                lda  #129
c123 8d 12 d4             sta  sid+18
c126 60         percex    rts
                ;
c127 d0 07      bass      bne  bass10  ;evtl.bass ->sid
c129 a9 2a                lda  #42     ;timing-fehler
c12b 20 d2 ff             jsr  chrout
c12e a9 ff                lda  #$ff
c130 30 24      bass10    bmi  bassex  ;pause
c132 a6 fb                ldx  rb
c134 f0 20                beq  bassex  ;->kein bass
c136 a2 20                ldx  #32
c138 8e 04 d4             stx  sid+4   ;vco#1 saegezahn+gate
c13b a2 40                ldx  #64
c13d 8e 0b d4             stx  sid+11  ;vco#2 rechteck+sync+gate
c140 8d 00 d4             sta  sid     ;vco#1 frequenz
c143 8c 01 d4             sty  sid+1
c146 8d 07 d4             sta  sid+7   ;vco#2 frequenz
c149 8c 08 d4             sty  sid+8
c14c a9 21                lda  #33
c14e 8d 04 d4             sta  sid+4
c151 a9 43                lda  #67
c153 8d 0b d4             sta  sid+11
c156 60         bassex    rts
                ;
                ;ton-parameter aus basic holen
                ;
c157 20 fd ae para        jsr  chkcom
c15a 20 9e b7             jsr  getbyt  ;h4
c15d 8e 1a c0             stx  save
c160 20 b2 b1             jsr  getpar  ;haupt-freq-wert
c163 a5 64                lda  $64
c165 d0 02                bne  par10
c167 a9 ff                lda  #$ff    ;aus null wird $ff
c169 8d 1b c0 par10       sta  save+1  ;   hb
c16c a5 65                lda  $65
c16e 8d 1c c0             sta  save+2  ;   lb
c171 20 b2 b1
c174 a5 64                lda  $64
c176 8d 1d c0             sta  save+3  ;   hb
c179 a5 65                lda  $65
c17b 8d 1e c0             sta  save+4  ;   lb
c17e ad 16 c0 par20       lda  fs      ;check freq-loc.frei
c181 f0 05                beq  par40   ;ja
c183 ad 15 c0             lda  h4
c186 d0 f6                bne  par20   ;warten wenn h4>0
c188 a2 04      par40     ldx  #4
c18a bd 1a c0 par60       lda  save,x  ;param.uebertragen
c18d 9d 15 c0             sta  h4,x
c190 ca                   dex
c191 10 f7                bpl  par60
                ;
c193 20 fd ae             jsr  chkcom  ;string holen
c196 20 9e ad             jsr  frmevl
c199 20 a3 b6             jsr  frestr
c19c aa                   tax
c19d a0 00                ldy  #0
c19f e8                   inx
c1a0 ca         par80     dex          ;string uebertragen
c1a1 f0 08                beq  par90   ;string zu ende
c1a3 b1 22                lda  ($22),y
c1a5 99 c3 c2             sta  string,y
c1a8 c8                   iny
c1a9 d0 f5                bne  par80
c1ab a9 00      par90     lda  #0      ;mit null abschliessen
c1ad 99 c3 c2             sta  string,y
c1b0 60                   rts
                ;
                ;naechsten ton im trend suchen
                ;
c1b1 20 3f c2 trend       jsr  getint  ;hole bit-muster ha%()
                ;
c1b4 8d 24 c0             sta  ha      ;l.b.
c1b7 8e 25 c0             stx  ha+1    ;h.b.
c1ba 20 3f c2             jsr  getint  ;hole trend tr%
c1bd 8d 29 c0             sta  tr
c1c0 20 3f c2             jsr  getint  ;hole ton-nr. ta%
c1c3 8d 2a c0 tre010      sta  ta
c1c6 ad 29 c0 tre020      lda  tr      ;ta+tr->ta
c1c9 20 65 c2             jsr  chkakk  ;check ob akkordeigen
c1cc f0 f8                beq  tre020  ;nein ->loop
c1ce 20 59 c2             jsr  putta   ;ta% absp.
c1d1 60                   rts
                ;
                ;uebergangston zu nae.harmonie suchen
                ;
c1d2 20 3f c2 hnext       jsr  getint  ;hole ha%(h)
c1d5 8d 24 c0             sta  ha
c1d8 8e 25 c0             stx  ha+1
c1db 20 3f c2             jsr  getint  ;hole ha%(hn)
```

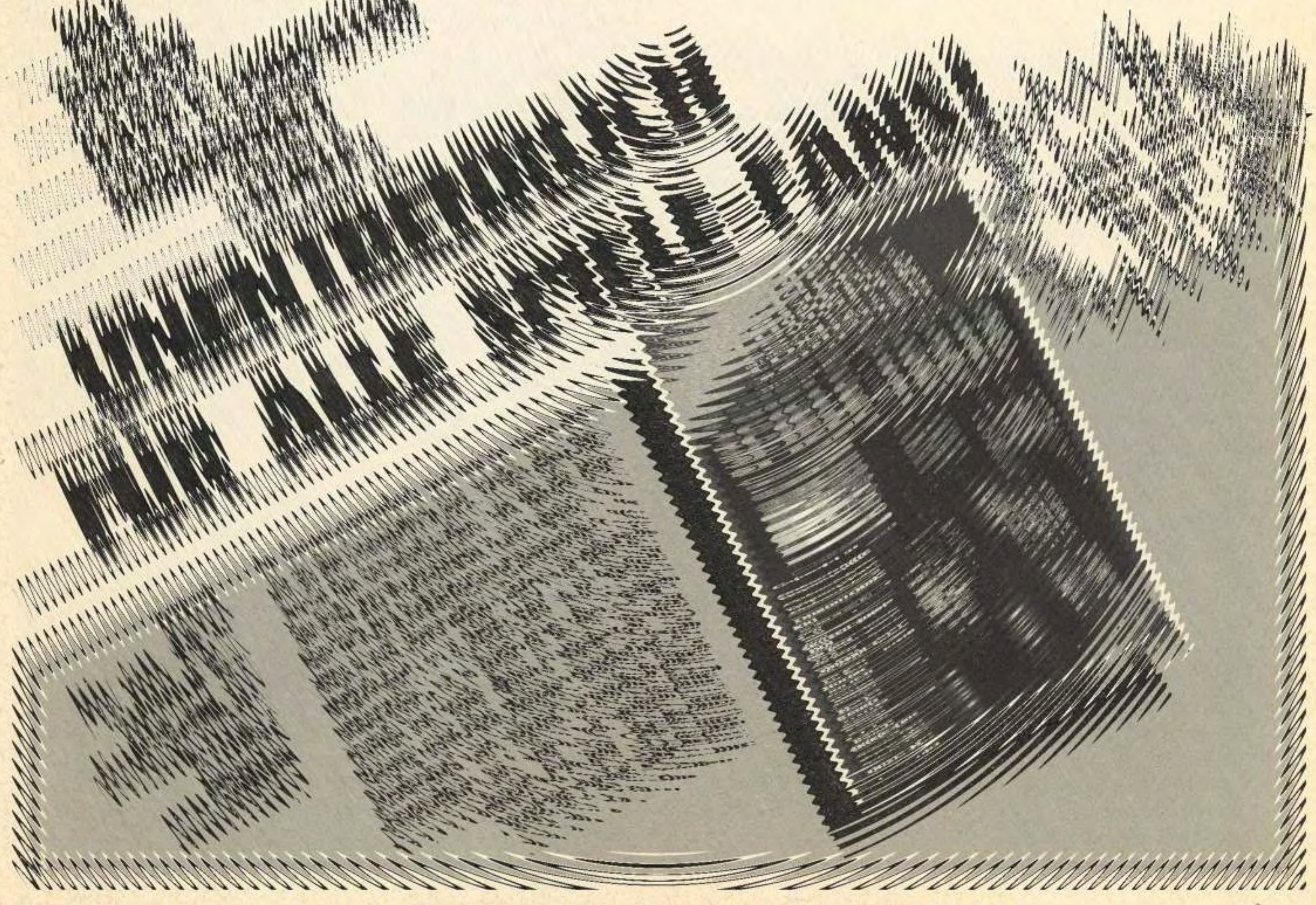
Listing 3. Das Quellcode-Listing zu »BASS/IRQ« (Fortsetzung)

64ER ONLINE
64er-online.de

```
c1de 8d 26 c0             sta  hanx
c1e1 8e 27 c0             stx  hanx+1
c1e4 20 3f c2             jsr  getint  ;hole hg%(h)
c1e7 8d 28 c0             sta  hg
c1ea 20 3f c2             jsr  getint  ;hole hg%(hn)
c1ed 8d 2a c0             sta  ta      ;->wird ta
c1f0 20 3f c2             jsr  getint  ;hole tr%
c1f3 8d 29 c0             sta  tr
c1f6 20 3f c2             jsr  getint  ;hole ta%
c1f9 8d 1a c0             sta  save    ;ta% saven
                ;1.var.  suche nachbar-ton v.nae.grundton, ...
                ;..der akkordeigen zu akt.harmonie ist
c1fc a9 ff                lda  #255    ;ta-1->ta (-1/2 ton)
c1fe 20 65 c2             jsr  chkakk  ;check ob akkordeigen
c201 d0 32                bne  hnexit  ;->ja, neuer ton gefunden
c203 a9 02                lda  #2      ;ta+2->ta (+1/2 ton)
c205 20 65 c2             jsr  chkakk  ;check ob akkordeigen
c208 d0 2b                bne  hnexit  ;->ja, neuer ton gefunden
c20a a9 fd                lda  #253    ;ta-3->ta (-1 ton)
c20c 20 65 c2             jsr  chkakk  ;check ob akkordeigen
c20f d0 24                bne  hnexit  ;->ja, neuer ton gefunden
c211 a9 04                lda  #4      ;ta+4->ta (+1 ton)
c213 20 65 c2             jsr  chkakk  ;check ob akkordeigen
c216 d0 1d                bne  hnexit  ;->ja, neuer ton gefunden
                ;2.var.  suche ton, der fuer beide harm. akkordeigen
c218 ad 24 c0             lda  ha
c21b 2d 26 c0             and  hanx
c21e 8d 24 c0             sta  ha
c221 ad 25 c0             lda  ha+1
c224 2d 27 c0             and  hanx+1
c227 8d 25 c0             sta  ha+1
c22a 0d 24 c0             ora  ha      ;check ob gemeins.toene
c22d d0 0a                bne  hne020  ;->ja
c22f ad 28 c0             lda  hg      ;nein, grundton nehmen
c232 8d 2a c0             sta  ta
c235 20 59 c2 hnexit      jsr  putta   ;ta% absp.
c238 60                   rts
                ;
c239 ad 1a c0 hne020      lda  save    ;ta% holen und laut trend...
c23c 4c c3 c1             jmp  tre010  ;...gemeins.akkord-ton suchen
                ;
                ;hole integer aus basic
                ;
c23f 20 fd ae getint      jsr  chkcom  ;komma
c242 20 8b b0             jsr  getvar  ;var.suchen
c245 85 49                sta  $49     ;var.adr. absp.
c247 84 4a                sty  $4a
c249 a5 0e                lda  $0e     ;check ob integer
c24b f0 09                beq  geterr  ;->nein, error
c24d a0 00                ldy  #0
c24f b1 49                lda  ($49),y  ;var.wert holen
c251 aa                   tax           ;h.b. ->reg.x
c252 c8                   iny
c253 b1 49                lda  ($49),y  ;l.b. ->reg.a
c255 60                   rts
                ;
c256 4c 99 ad geterr      jmp  typerr  ;error
                ;
                ;ta% als basic-integer-var. absp.
                ;
c259 a9 00    putta       lda  #0
c25b a8                   tay
c25c 91 49                sta  ($49),y  ;h.b.
c25e ad 2a c0             lda  ta
c261 c8                   iny
c262 91 49                sta  ($49),y  ;l.b.
c264 60                   rts
                ;
                ;check ob ton nr.(ta)+reg.a  = akkordeigen
                ;in  reg.a=inkr./dekr. auf ta
                ;
c265 18       chkakk      clc
c266 6d 2a c0             adc  ta      ;ta+inkr/dekr ->ta
c269 10 03                bpl  cak010  ;check ob ta im bereich 0...11
c26b 18                   clc
c26c 69 0c                adc  #12     ;...sonst-korrektur
c26e c9 0c    cak010      cmp  #12
c270 90 03                bcc  cakbit
c272 38                   sec
c273 e9 0c                sbc  #12
                ;
c275 8d 2a c0 cakbit      sta  ta      ;bit f.akt.ton holen
c278 aa                   tax
c279 bd 2b c0             lda  mask,x  ;and-maske holen
c27c a0 00                ldy  #0
c27e e0 08                cpx  #8      ;check ob l.b. oder h.b
c280 90 02                bcc  cak030  ;->l.b.
c282 a0 01                ldy  #1      ;h.b.
c284 39 24 c0 cak030      and  ha,y    ;bit aus akt.harm.extrahieren
c287 60                   rts
                ;
                ; zufalls-ton ermitteln
                ;
c288 20 3f c2 zufall      jsr  getint  ;hole ha%(h)
c28b 8d 24 c0             sta  ha      ;l.b.
c28e 8e 25 c0             stx  ha+1    ;h.b.
c291 20 3f c2             jsr  getint  ;hole ta%
c294 20 75 c2             jsr  cakbit  ;bit f.akt.ton holen
c297 49 ff                eor  #$ff    ;...und loeschen
c299 39 24 c0             and  ha,y    ;...damit nicht nochmals
c29c 99 24 c0             sta  ha,y    ;...gleicher ton kommt.
c29f ad 12 d0             lda  random  ;zufalls-zahl + ta ->ta
c2a2 29 07                and  #7
c2a4 d0 02                bne  zuf030
c2a6 a9 01    zuf020      lda  #1
c2a8 20 65 c2 zuf030      jsr  chkakk  ;check ob akk.eigen
c2ab f0 f9                beq  zuf020  ;->nein, weiter suchen
c2ad 4c 59 c2             jmp  putta   ;ja, ta% als basic-var.absp.
                ;
                ; tempo aus basic holen
                ;
c2b0 20 3f c2 tempo       jsr  getint  ;hole t2% (1.vorschlag)
c2b3 8d 1f c0             sta  t2
c2b6 20 3f c2             jsr  getint  ;hole t3% (2.vorschlag)
c2b9 8d 20 c0             sta  t3
c2bc 20 3f c2             jsr  getint  ;hole t4% (1/4-hauptschlag)
c2bf 8d 21 c0             sta  t4
c2c2 60                   rts
                ;
c2c3          string      =    *
```

Listing 3. Das Quellcode-Listing zu »BASS/IRQ« (Schluß)

64ER ONLINE



Alle Tasten-, Zeichen- und Steuercodes

Die gute und übersichtliche Tastatur der Computer von Commodore bietet gerade dem Anfänger sehr leichte Einstiegsmöglichkeiten. Wer aber tiefer in die Computerei eindringt, stößt beim VC 20 und Commodore 64 schnell auf eine verwirrende Vielfalt von Codes und Steuerzeichen, deren Kenntnis und Beherrschung jedoch auch dem Anfänger ein weites Anwendungsfeld eröffnen kann.

Haben Sie schon einmal einen Ball länger als ein paar Minuten auf derselben Stelle liegen sehen? Ich noch nicht, denn der nächste Vorbeikommer kickt ihn – irgendwohin. Sie doch auch, oder ? Dasselbe Phänomen können Sie überall dort beobachten, wo ein Heimcomputer steht. Eingeschaltet oder nicht – jeder der vorbeikommt, drückt auf eine Taste. Und wenn das Ding gar reagiert – mit einem Buchstaben auf dem Bildschirm – dann bleibt selbst die Oma stehen und tippt nochmal und nochmal.

Die Tasten wirken auf die Menschen wie das Licht auf die Motten

Wenn vom VC 20 oder Commodore 64 die Rede ist, kommt dieselbe sehr rasch auf die Tastatur. Ihre Schreibmaschinen-Qualität und die klaren Steuer- und Funktionstasten heben sie aus der Schar der Konkurrenten heraus. Sie ist sehr benutzerfreundlich, besonders für Anfänger, während der ersten Tast(en)versuche.

Aber rasch wird offensichtlich, daß die Tasten auch ihre Geheimnisse haben, und zwar frühestens, wenn die vier Funktionstasten keine Reaktionen, zumindest keine sichtbaren, hervorrufen, und spätestens bei dem Versuch, in einem Programmablauf bestimmte Tastenfunktionen ausführen zu lassen, wie zum Beispiel »Cursor Down« oder »Bildschirm löschen«.

Ich will Ihnen deshalb in diesem Aufsatz die Grundlagen der Tastaturabfrage, das Geheimnis der Funktionstasten, die CHR$-Anhängsel von PRINT-Befehlen, die mysteriösen »negativen« Zeichen in Anführungsstrichen und gleichzeitig gedrückte Tasten erklären und mit Kochrezepten ausschmücken.

Übrigens: das hier Gesagte gilt sowohl für den VC 20 als auch für den Commodore 64. Nur die Programmbeispiele nicht, die sind auf den kleinen VC 20 zugeschnitten. Die 64er mögen mir das verzeihen. Ich scheue mich aber, Kochrezepte abzugeben, die ich nicht selbst ausprobiert habe. Und ich habe leider (noch) keinen C 64. Falls Unterschiede bei den Adressen auftreten, gebe ich den Wert für den C 64 in Klammern an. Als erstes möchte ich folgende Behauptung aufstellen:

Dem Computer sind alle Tasten gleich.

Und er behandelt sie auch alle gleich. 60mal in jeder Sekunde unterbricht der Computer was er auch immer gerade ausführt, merkt sich, wobei und wo er gerade unterbrochen hat, und schaut nach, ob eine der Tasten gedrückt worden ist.

Keine Regel ohne Ausnahme, auch beim Computern nicht: Die RESTORE-Taste und die SHIFT-LOCK-Taste fallen völlig aus dem Rahmen und haben mit den kommenden Erklärungen nichts zu tun. Ich werde sie deshalb auch nicht mehr erwähnen. Der Vollständigkeit halber sei hier nur kurz gesagt, daß die RESTORE-Taste den Computer bei jeglicher Arbeit unterbricht und ihn mit READY und blinkendem Cursor in den Anfangszustand zurücksetzt. Die SHIFT-LOCK-Taste ist (wie bei der Schreibmaschine) eine mechanische Arretierung der SHIFT-Taste. Es bleiben uns immerhin 64 Tasten, die vom Computer während seiner Verschnaufpause inspiziert werden. Die Funktionstasten sind immer dabei! Diese Inspektion – Tastaturabfrage genannt – wollen wir uns näher anschauen.

Der Computer erhält nicht, wie es eigentlich logisch wäre, von jeder gedrückten Taste ein spezielles Code-Signal. Das wäre für einen Heimcomputer zu aufwendig und zu teuer. Das Betriebssystem des Computers veranstaltet vielmehr eine Befragung seines Tastenvolkes, nach dessen Stimmenabgabe er dann entscheidet, welche Taste nun eigentlich gedrückt worden ist. Schauen Sie sich bitte das Bild 1 genauer an. Es zeigt Ihnen die 64 Tasten in einer 8 x 8-Matrix. Diese Anordnung entspricht der elektrischen Verbindung der Tasten. Die Abfrage selbst ist sehr einfach. Der Computer ruft alle senkrechten Spalten einzeln auf, indem er die Zahl, die an der jeweiligen Spalte in Bild 1 steht, in eine spezielle Speicherzelle hineinschiebt. Beim VC 20 ist das 37152, beim Commodore 64 die 56320. Falls die gedrückte Taste in dieser Spalte liegt, meldet sie sich und die Zahl der waagrechten Zeile wird in die Adresse 37153 (56321) geschrieben. Wenn überhaupt keine Taste gedrückt war, erscheint in 37153 (56321) eine 255.

10 IF Sie jetzt fragen, warum die Zahlen so willkürlich und außerdem paarweise gleich sind, THEN lesen Sie weiter: GOTO nächsten Absatz.

20 REM ein bißchen Theorie würde nicht schaden: GOTO übernächsten Absatz.

Die Zahlen sind natürlich nicht willkürlich gewählt. Es sind vielmehr die Dezimalwerte von Dualzahlen, die beim Anwählen einer Spalte beziehungsweise durch Drücken einer Taste in den schon genannten Speicherzellen 37152 (56320) und 37153 (56321) entstehen. Bild 2 zeigt den Zusammenhang.

Am Rande der Matrix sehen Sie jetzt nicht die Zahlen, sondern die beiden Speicherzellen, im folgenden »Register« genannt, und ihre bitweise Verbindung mit Spalten und Zeilen der Tasten. Prinzipiell gilt, daß diejenige Spalte angewählt ist, an deren Stelle eine »Null« im Register 37152 (56320) steht.

Theorie und Praxis: eine untrennbare Einheit

Das ergibt im Register eine Dualzahl, die entsprechend der Bit-Numerierung zu lesen ist. Eine gedrückte Taste ihrerseits erzeugt eine Null im Register 37153 (56321) an der Stelle, wo ihre Zeile angeschlossen ist. Ich habe im Bild 2 ein Beispiel (Taste mit den Zeichen »/«) eingezeichnet, und wenn Sie die Dualzahlen kennen, werden Sie beim Vergleich mit den Zahlen in Bild 1 die Übereinstimmung erkennen.

Wie bitte? Sie sagen, daß es aber doch möglich sein müßte, weit mehr als die im Bild 1 gezeigten acht Zahlen im Register 37153 (56321) zu erzeugen, indem man mehr als eine Taste gleichzeitig drückt? Richtig gesehen, das geht in der Tat, und

64er-online.de

das wollen wir uns auch gleich einmal ansehen. Dazu brauchen wir ein Kochrezept. Ich schlage vor, das Rezept - und auch die noch folgenden - gleich auszuprobieren. Nehmen Sie diesen Artikel und verfolgen Sie die weiteren Zeilen meiner Beschreibung abwechselnd lesend und eintippend.

Die Abfrage der Tastatur wird auf dem Bildschirm dargestellt

Als erstes wollen wir, so wie der Computer es macht, eine Spalte anwählen, zum Beispiel wie im Bild 2, die vierte von rechts. Das ergibt die Dualzahl 11110111, in dezimal die Zahl 247.

```
100 POKE 37152,247
200 PRINT PEEK(37152),PEEK(37153)
300 GOTO 100
```

Zeile 200 druckt uns sowohl den von uns gePOKEten Wert 247 als auch den in Register 37153 (56321) stehenden Wert aus, der dort erscheint, sobald wir eine Taste in der Spalte 247 drücken. Der Rücksprung in Zeile 300 erzeugt auf dem Bildschirm zwei senkrechte, durchlaufende Zahlenstreifen.

Nach RUN läuft rechts die Zahl 255 (dual = lauter 1er, das heißt, keine Taste gedrückt). Drücken Sie nun die »/«-Taste und es erscheint die 191. Die X-Taste erzeugt 251, die links angeordnete SHIFT-Taste eine 253. Die rechte SHIFT-Taste dagegen zeigt keine Reaktion - ist auch klar, denn sie liegt ja in einer anderen Spalte.

Aber es ist interessant, sich zu merken, daß bei einer Tastaturabfrage die beiden SHIFT-Tasten völlig eigenständig behandelt werden, im Gegensatz zu ihrer Funktion. Sie sehen, ich habe recht gehabt: Alle Tasten sind gleich.

Sie können auch die RUN-STOP-Taste drücken, aber bitte nur ganz, ganz kurz antippen, denn das Programm bleibt natürlich entsprechend ihrer ungeSHIFTteten Funktion sofort stehen. Wenn das Betätigen der Taste aber kurz genug war, dann steht als letzte Zahl die 254 da.

Gleichzeitige Abfrage von zwei Tasten

Jetzt kommt das große Experiment. Nach neuem RUN drücken Sie »Cursor down« und »/« gleichzeitig - es erscheint 63. Der Blick auf Bild 2 zeigt uns für beide Tasten das Bitmuster 00111111 und das ist 63. Es geht also. Probieren Sie ruhig alle Kombinationen aus. Wenn Sie Zeit und die Gelenkigkeit eines Konzertpianisten haben, können Sie alle Zahlen bis 254 erzeugen, auch die Null. Dazu sind alle acht Tasten gleichzeitig zu drücken.

Erwähnenswert ist, daß jetzt das Programm nicht stehenbleibt, obwohl die STOP-Taste gedrückt ist. Aber die SHIFT-Taste ist auch gedrückt, und das ist laut Handbuch die LOAD-RUN-Funktion, die jetzt nicht zum Tragen kommt.

Sie wollen sicher endlich auch einmal die Funktionstasten zur Geltung bringen. Dazu müssen wir aber eine andere Spalte anwählen. Ändern Sie bitte die Zeile 100 ab:

```
100 POKE 37152,239
```

In der Spalte 239 liegt die F1-Taste. Nach RUN passiert aber etwas Komisches: Trotz des POKEns von 239 läuft links wieder die 247, genauso wie vorhin. Und die Tasten der Spalte 239 reagieren nicht, nur die der Spalte 247.

Was passiert da? Es liegt daran, daß nur beim Eintippen eines Programms die Tastaturabfrage genauso funktioniert wie beschrieben. Wenn aber ein Programm abläuft, dann ist der Computer nur an der STOP-Taste interessiert und schiebt deshalb immer wieder eine 247 in das Register 37152 (56320).

Eine Ausnahme gibt es auch hier: INPUT- und GET-Befehle fragen auch die anderen Tasten ab.

Diese erzwungene Vorfahrt der Spalte 247 können wir durch unsere Zeile 100 nicht ändern, denn wir sind ja nach RUN in einem Programmablauf. Die einzige Möglichkeit, andere Spalten POKEn zu können, ergibt sich für uns durch Programmierung in Maschinensprache. Aber darauf will ich nicht jetzt schon, sondern erst am Schluß des Aufsatzes eingehen.

Wir haben also bei unserem Versuch, das Ergebnis der Befragung schon bei der Stimmenabgabe - sozusagen im Wahllokal - auszukundschaften, Pech gehabt. Das macht aber nichts, denn irgendwann wird ja ein Wahlergebnis offiziell bekanntgegeben. Bei der Tastaturabfrage ist das auch so. Vorher aber wollen wir wenigstens aus dem Teilergebnis, das wir ausspioniert haben, Kapital schlagen und die mehrfache Tastenabfrage der Spalte 247 an einem klaren Beispiel demonstrieren.

Programmsteuerung mit zwei unabhängigen Tasten

Das Programm soll auf dem Bildschirm in den Spalten 6 und 15 zwei senkrechte Bänder mit Sternen darstellen, deren Farbe mit der »X«- und der »/«-Taste unabhängig voneinander, auch gleichzeitig, verändert werden kann.

```
100 PRINT CHR$(147)
```

Diese Zeile löscht den Bildschirm. Die Methode (ASCII-CODE) kennen Sie sicher aus dem Programmier-Handbuch. Ich werde sie aber noch genau behandeln.

```
110 BS=PEEK(648)*256
120 FS=4*(PEEK(36866) AND 128) + 37888
```

Zeile 110 und 120 machen das Programm unabhängig von Speichererweiterungen. BS und FS sind Variable für die Anfangsadressen des Bildschirm- beziehungsweise des Farbspeichers.

```
130 F=2:G=2
```

Das ist die Anfangsfarbe »Rot« für beide Sternreihen.

```
200 FOR Z=0 TO 22
```

Mit Z werden die 23 Zeilen abgezählt.

```
300 POKE BS+5+Z*22,42
310 POKE FS+5+Z*22,F
```

Keine Angst, ich mute Ihnen keine höhere Mathematik zu. Ab Zeile 300 wird der Bildschirm-Code (42) für den Stern (auch das behandle ich später noch) in Spalte 6 (BS+5) für jede Zeile (Z*22) untereinander gePOKEt, dazu die Farbe F in den gleichen Platz des Farbspeichers. Dasselbe gilt in Zeile 400 und 410 für die Spalte 15.

```
400 POKE BS+15+Z*22,42
410 POKE FS+15+Z*22,G
500 AA=PEEK(37153)
510 IF AA=191 OR AA=187 THEN F=F+1
520 IF AA=251 OR AA=187 THEN G=G+1
```

Ab Zeile 500 werden die »X«-Taste (191) und die »/«-Taste (251) abgefragt. Wenn eine davon gedruckt ist, wird die Farbzahl F beziehungsweise G um 1 erhöht. Die OR-Funktion, mit welcher der Wert 187 abgefragt wird, erlaubt ein gleichzeitiges Drücken beider Tasten, es werden sowohl F als auch G erhöht.

Um die Farben zwischen 2 (Rot) und 7 (Gelb) zu begrenzen, verwenden wir:

```
600 IF F=8 THEN F=2
610 IF G=8 THEN G=2
```

Zum Schluß wird der Zeilenzähler Z weitergesetzt. Wenn er 22 (das heißt die 23. Zeile) erreicht hat, springt das Programm an den Anfang zurück.

```
700 NEXT Z
800 GOTO 200
```

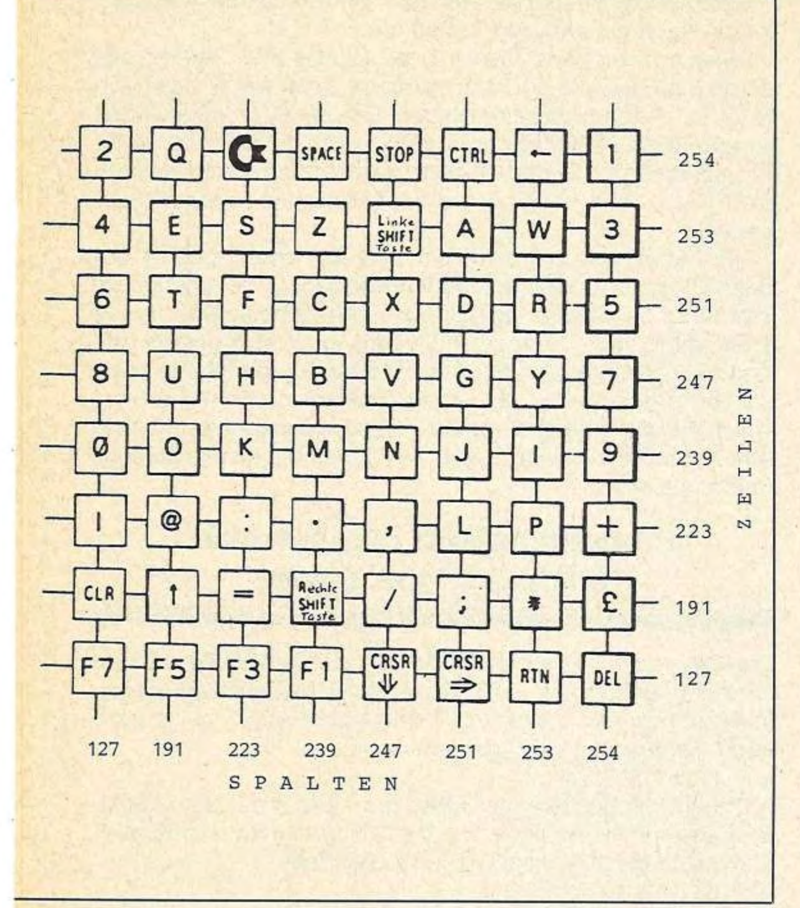


Bild 1. Die »elektrische« Anordnung der Tasten

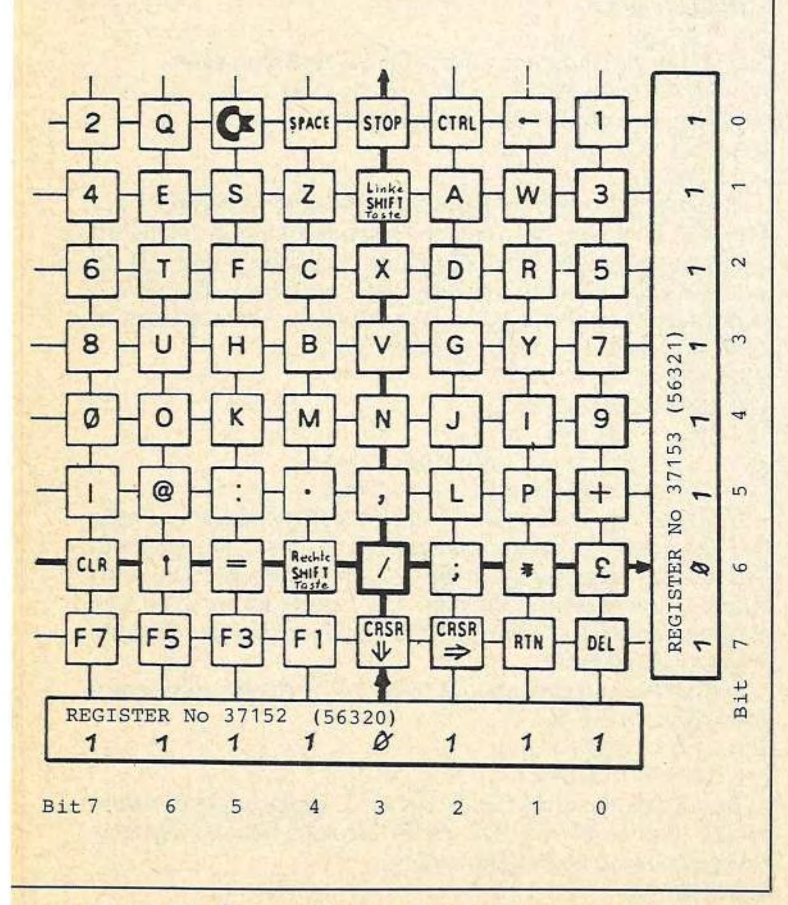


Bild 2. Die Tasten-Matrix mit ihren Registern

Nun geben Sie RUN ein. Wenn Sie eine der Tasten zu lange, das heißt länger als einen Z-Zyklus, drücken, springt die Farbe weiter. Um das etwas zu erleichtern, können Sie noch eine Verzögerung einbauen:

```
220 FOR T=1 TO 50: NEXT T
```

Ich gebe ja zu, das ist kein gewaltiges Programm. Aber es zeigt Ihnen wenigstens, wie auch in Basic eine mehrfache Tastenabfrage möglich ist.

```
100 PRINT CHR$(147)
110 BS=PEEK(648)*256
120 FS=4*(PEEK(36866) AND 128) + 37888
130 f=2:g=2
200 FOR Z=0 TO 22
220 FOR T=1 TO 50: NEXT T
300 POKE BS+5+Z*22,42
310 POKE FS+5+Z*22,F
400 POKE BS+15+Z*22,42
410 POKE FS+15+Z*22,G
500 AA=PEEK(37153)
510 IF AA=191 OR AA=187 THEN F=F+1
520 IF AA=251 OR AA=187 THEN G=G+1
600 IF F=8 THEN F=2
610 IF G=8 THEN F=2
700 NEXT Z
800 GOTO 200
```

Weiter geht's, und zwar mit dem schon erwähnten Bekanntgeben des Wahlergebnisses. In anderen Worten: Wie wertet der Computer die Tastenabfrage über die Register 37152 (56320) und 37153 (56321) weiter aus?

Sobald der Computer merkt, daß eine Taste gedrückt ist, nimmt er die beiden Zahlen, die in den Registern 37152 (56320) und 37153 (56321) stehen, und wandelt sie in eine Code-Zahl um, die er in der Speicherzelle 203 ablegt. Die Code-Zahl steht auch in der Speicherzelle 197. Mit dem Grund für diese Verdoppelung muß ich mich aber erst noch beschäftigen.

Wir bleiben bei Adresse 203. Wie bei der Abfrage der Tastatur-Matrix wollen wir uns den Inhalt dieser Speicherzelle ansehen. Löschen Sie bitte das alte Programm und geben Sie ein:

```
100 PRINT PEEK(203)
200 GOTO 100
```

Nach RUN sehen wir wieder die laufende Zahlenbank, jetzt aber mit 64. Das ist die Code-Zahl für »keine Taste gedrückt«. Die »/«-Taste ergibt jetzt 30, die X-Taste 26 und so weiter. Endlich ist es soweit! Die Funktionstasten reagieren und geben ihre Code-Zahl preis.

Die Funktionstasten reagieren doch!

Probieren Sie alle Tasten durch und schreiben Sie die Code-Zahlen auf die Tasten von Bild 1 oder Bild 2. Jetzt sehen Sie auch die Gesetzmäßigkeit, nach der der Computer die Spalten- und Zeilenzahl der Register ummodelt. Schreiben Sie sich am besten eine komplette Liste der Code-Zahlen für die weitere Verwendung. Die RUN-STOP-Taste läßt sich hier leichter als beim ersten Mal überlisten, natürlich nur mit gleichzeitigem SHIFT.

Apropos »gleichzeitig«! Wiederholen Sie das Experiment von vorhin. Hier erleiden wir unseren zweiten Fehlschlag: Mehrfachtasten geben keinen Sinn, denn die Umcodierung verwehrt es uns. Wie gut, daß wir die Methode der Matrix-Abfrage haben, auch wenn sie in voller Eleganz nur in Maschinensprache möglich ist. Doch wie gesagt, davon später.

Zurück zu den einzelnen Tasten.

In der Liste der Code-Zahlen fehlen die Tasten RESTORE, SHIFT, C=, CTRL. Sie haben das nicht bemerkt? Dann haben

Sie auch noch nicht die von mir vorgeschlagene Liste gemacht!

Um auch diese Zahlen auf den Bildschirm zu bringen, ergänzen Sie bitte die Zeile 100 in

```
100 PRINT PEEK(203,PEEK(653)
200 GOTO 100
```

Jetzt sehen Sie zwei Zahlenreihen laufen. Zu der bekannten Reihe ist auf der zweiten Hälfte des Bildschirms (bedingt durch das Komma zwischen den PEEKs) eine 0-Reihe gekommen. Drücken Sie jetzt die SHIFT-Taste: Rechts läuft eine 1 – die Code-Zahl dieser Taste.

Die Steuertasten haben ihre Code-Zahl in der Speicherzelle 653

Die C=-Taste erzeugt eine 2, die CTRL-Taste eine 4 (und die natürlich ganz langsam). Drücken Sie mal SHIFT und C= gleichzeitig. Siehe da, bei der Speicherzelle 653 und ihren Steuertasten gibt das einen Sinn. Die Tabelle aller Kombinationen sieht so aus:

keine Taste	0
SHIFT	1
C=	2
SHIFT und C=	3
CTRL	4
SHIFT und CTRL	5
CTRL und C=	6
alle drei Tasten	7

Während wir das ausprobiert haben, läuft links unbeirrt die 64. Und in der Tat, durch das Aufspalten und Abspeichern der Code-Zahlen in zwei getrennte Speicherzellen können wir beide Tastenarten, Zeichentasten und Steuertasten, unabhängig voneinander und/oder gleichzeitig abfragen.

Das nutzen wir zum Beispiel bei den Funktionstasten aus. Jede von ihnen hat ihre eigene Code-Zahl in Adresse 203. Aber das gäbe uns nur vier Möglichkeiten, entsprechend der Aufschrift der ungeraden f-Zahlen.

Um auch f-2 bis f-8 zu erhalten, kombinieren wir die vier Zahlen in 203 einfach mit SHIFT-Taste gedrückt oder nicht (1 oder 0 in 653).

Aber Sie sehen schon, wie willkürlich das ist, denn wir könnten f-6 auch definieren als Kombination von der dritten Funktionstaste und CTRL (also 55 und 4).

Überhaupt, wir sind gar nicht auf acht Funktionstasten beschränkt, wie es uns durch den Ausdruck eingeredet wird. Die vier Funktionstasten ergeben zusammen mit den acht Kombinationen der Steuertasten 32 mögliche Funktionen. Natürlich gilt das für alle Tasten der Tastatur. Der Computer selbst nutzt allerdings nur wenige Kombinationen aus. SHIFT und C= (3) schaltet alle Buchstaben in Groß-/Kleinschrift um, die CTRL-Taste mit den Zahlentasten erzeugt die Vordergrund-Farben. Sie haben also viel Raum für phantasievolle Abfragekombinationen. Die Abfrage selbst und ihre Verwendung in einem Programm will ich abschließend mit den Funktionstasten demonstrieren.

Eine »Heimorgel« ganz besonderer Art

Wie man mit dem VC 20 Töne erzeugt, wissen Sie. In Zeile 10 definieren wir das Sopranregister und geben ihm den Namen Z, in Zeile 20 schalten wir die Lautstärke ein. Lautstärke des Fernsehers nicht vergessen!

```
10 Z=36876
20 POKE 36878,10
```

Ab Zeile 40 bis 110 wird jede einzelne Kombination der Code-Zahlen von Funktions- und Steuertasten abgefragt. Sobald sie auftritt, wird ein entsprechender Ton der Tonleiter gePOKEt.

```
30 A=PEEK(203): B=PEEK(653)
40 IF A=39 AND B=0 THEN POKE Z,131
50 IF A=47 AND B=0 THEN POKE Z,145
60 IF A=55 AND B=0 THEN POKE Z,157
70 IF A=63 AND B=0 THEN POKE Z,162
80 IF A=39 AND B=1 THEN POKE Z,172
90 IF A=47 AND B=2 THEN POKE Z,181
100 IF A=55 AND B=4 THEN POKE Z,189
110 IF A=63 AND B=7 THEN POKE Z,193
```

Ein Rücksprung in die Zeile 30 verleiht dem Ton auch die Dauer.

```
120 GOTO 30
```

Damit ein Ton aber nur so lange klingt, wie eine Tastenkombination gedrückt ist, schieben wir noch Zeile 35 ein, die das Sopranregister auf 0 (Stille) setzt, sobald »keine Taste« gedrückt ist.

```
35 IF A=64 AND B=0 THEN POKE Z,0
```

Jetzt will ich Ihnen natürlich noch die von mir gewählten Tastenkombinationen verraten, damit Sie gezielt »Alle meine Entchen« spielen können. Es gilt der Reihe nach:

Zeile 40	f-1
Zeile 50	f-3
Zeile 60	f-5
Zeile 70	f-7
Zeile 80	f-1 und SHIFT
Zeile 90	f-3 und C=
Zeile 100	f-5 und CTRL
Zeile 110	f-7 und CTRL und C= und SHIFT

Ich gebe zu, daß diese Tastenauswahl nicht gerade eine bequeme Klaviatur ergibt.

Ihrem Ehrgeiz ist es überlassen, eine Orgel zu programmieren, die zwar immer noch einstimmig ist, aber eine »normale« Klaviatur hat und auch den vollen Tonumfang des VC 20 ausnutzt. Stellen Sie einfach die Code-Zahlen der Zeichentasten so zusammen, daß eine Tastenreihe die »weißen« Tasten und die darüberliegende Reihe die »schwarzen« Tasten darstellt.

Tastenabfrage und kein Ende: was es sonst noch alles gibt

Mit den Steuertasten können Sie die Oktaven umschalten, mit den Funktionstasten verschiedene Lautstärken. Die Zahlen für die Abfrage entnehmen Sie der Liste.

Statt Töne zu POKEn, können Sie natürlich mit dieser Abfragetechnik der Funktionstasten (und auch der anderen Tasten) alles mögliche per Programm steuern: Raumschiffe abschießen, Textseiten weiterschalten, den Hund rauslassen oder Toast rösten.

Ich habe Ihnen bisher gezeigt, wie alle Tasten des VC 20 beziehungsweise des C 64 in einer Matrix angeordnet sind. Sobald eine Taste gedrückt wird, steht eine spezielle, nur dieser einen Taste zugeteilte Code-Zahl im Register 37152 des VC20. Das entsprechende Register des C64 ist 56320.

Auch habe ich erklärt, wie die Code-Zahlen zustande kommen. In einem kleinen Demonstrationsprogramm haben wir dann durch Abfragen dieses Registers mit PEEK bestimmte Programmschritte mit Tastendruck gesteuert.

Wir haben herausgefunden, daß das Betriebssystem des Computers bei Verwendung von Basic eine Abfrage von nur acht Tasten zuläßt. Die Abfrage aller Tasten, auch mehrerer Tasten gleichzeitig, mit einem Programm in Maschinencode habe ich Ihnen für den Schluß versprochen.

```
LIST

10 PRINT CHR$(65)
20 PRINT CHR$(156)
30 PRINT CHR$(66)CHR$(13)CHR$(67)
99 END
READY.
RUN
A

B
C
READY.
```

Bild 3. Auf dem Bildschirm haben »B« und »C« eine andere Farbe wie »A«

```
LIST

10 PRINT CHR$(65)
20 PRINT CHR$(156)
30 PRINT CHR$(66)CHR$(17)CHR$(67)
99 END
READY.
RUN
A

B
 C
READY.
```

Bild 4. Resultat der Steuerfunktion 17, «CURSOR DOWN«

```
LIST

5 POKE 198,5
10 POKE 631,65
20 POKE 632,156
30 POKE 633,66:POKE 634,17:POKE 635,67
99 END
READY.
RUN

READY.
AB
  C
```

Bild 5. Die ASCII-WERTE wurden in den Tastaturpuffer gePOKEt

```
LIST

5 POKE 198,10
10 POKE 631,65:POKE 632,13
20 POKE 633,156
30 POKE 634,66:POKE 635,17:POKE 636,67
99 END
READY.
RUN

READY.
A

?SYNTAX  ERROR
READY.
B
 C
```

Bild 6. Das Beispielprogramm mit dem Codewert für RETURN

Wir sind aber noch einen Schritt weitergegangen und haben herausgefunden, daß diese Code-Zahlen der 64 Tasten umgerechnet und für Zeichen- und Steuertasten getrennt in die Speicherzellen 203 und 653 gebracht werden.

Mit dem folgenden kleinen Programm haben wir dann diese beiden Speicherzellen abgefragt und eine Tabelle (Tabelle 1) angefertigt.

Tippen Sie ein:

```
100 PRINT PEEK(203),PEEK(653)
200 GOTO 100
```

Der Grund, daß die Zahlen für die beiden Computer, trotz gleicher Tastatur, verschieden sind, liegt darin, daß die elektrische Anordnung der Tasten in einer 8 x 8-Matrix (die ich für den VC 20 gezeigt habe) beim C 64 anders sind.

Nun lesen Sie, wie man eine vollständige Tabelle der ASCII-Codes vom VC 20 und Commodore 64 erstellt. Außerdem werden Sie erfahren, daß die beiden Systeme wesentlich mehr Funktionstasten bieten, als Sie bisher vielleicht angenommen haben.

Die Wiederholung ist die Mutter der Weisheit

Im Gedenken an diesen Spruch meines Latein-Lehrers zeige ich noch einmal die Anwendung dieser Code-Zahlen im Listing 1. Zeilen in ( ), wie gesagt, sind für den C 64, aber nur dort, wo er sich vom VC 20 unterscheidet.

Natürlich wähle ich wieder die Funktionstasten und am besten auch noch eine andere Tastenkombination. Zweck der kleinen Demonstration soll das Umschalten auf verschiedene Bildschirmrahmen- und Hintergrundfarben sein. Ich schlage vor, Sie nehmen wie üblich die Zeitschrift zum Computer und lesen tippend weiter. Es folgt nun das Programm 1 zur Tastaturabfrage.

```
10 PRINT CHR$(147)
```

Diese Befehlsfolge löscht den Bildschirm. Die Code-Zahlen der Funktion CHR$ werde ich noch erklären.

Die Zeilen 20 und 30 erleichtern die Tipperei und machen übrigens das Programm ein bißchen schneller. Sie ordnen den Variablen A und B den Inhalt der Speicherzellen 203 und 653 – die wir ja abfragen wollen – zu.

```
20 A=PEEK(203)
30 B=PEEK(653)
```

Jetzt geht's los mit der Fragerei. Die Taste f1, mit der wir die Farbkombination Blau/Gelb schalten wollen, hat folgende Code-Zahlen:

```
40 IF A=39 AND B=0 THEN POKE 36879,126
(40 IF A=4 AND B=0 THEN POKE 53280,6:POKE 53281,7 )
```

f2 ist dieselbe Taste, aber geSHIFTet (B=1). Die Farben sollen jetzt Rot/Grün sein:

```
50 IF A=39 AND B=1 THEN POKE 36879,45
(50 IF A=4 AND B=1 THEN POKE 53280,5: POKE 53281,2)
60 IF A=47 AND B=4 THEN POKE 36879,25
(60 IF A=5 AND B=4 THEN POKE 53280,1: POKE 53281,1)
```

Zeile 60 bestimmt ebenfalls eine Funktionstaste und zwar f3. Allerdings habe ich sie willkürlich mit der CTRL-Taste kombiniert, nicht, um Sie zu verwirren, sondern um zu zeigen, daß wir mit den f-Tasten mehr als acht Funktionen festlegen können, nämlich 32! (Vier f-Tasten mal acht Steuertasten-Kombinationen).

Mit dem Klammeraffen »@« schalten wir die Farben in den Normalzustand zurück.

```
70 IF A=53 AND B=0 THEN POKE 36879,27
(70 IF A=46 AND B=0 THEN POKE 53280,3: POKE 53281,1)
```

Zeile 80 läßt das Programm im Kreis laufen, so daß die Funktionstasten beliebig oft ausprobiert werden können.

```
80 GOTO 020
```

Funktionstasten im Überfluß

So, jetzt kommt der absolute Hit dieser Methode!

Ich habe gerade vorher gesagt, daß wir nicht acht, sondern 32 mögliche Funktionstasten haben, nämlich durch Verwen-

64ER ONLINE

dung der vier f-Tasten mit den acht Steuertasten-Codes in Speicherzellen 653.

Dasselbe gilt für jede andere Taste natürlich auch!

Der Computer benützt allerdings einige davon, zum Beispiel die 1 (SHIFT) für die Zeichen über den Zahlen beziehungsweise rechts unten auf den Tasten, die 2 (C=) für die Zeichen links unten auf den Tasten und die 4 (CTRL) für die Farben.

Wir können daher mit den vom Computer benutzten Zahlen 3, 5, 6 und 7 aus Zelle 653 in Kombination mit allen 55 Tasten eine riesige Anzahl verschiedener »Funktionstasten« erfinden und sie in unseren Programmen einsetzen.

Der Computer geht bei der Abfrage noch einen Schritt weiter

Aber auch die 4 der CTRL-Taste, die ja nur die obere Reihe der Tasten beeinflußt, hat einen praktischen Wert, meiner Meinung nach sogar einen sehr großen, da sie ja nur einen einzigen Tastendruck erfordert und nicht eine Kombination.

Die CTRL-Taste kombiniert mit den 35 Tasten der unteren drei Reihen der Tastatur gibt uns mehr »Funktionstasten« als wir wahrscheinlich jemals brauchen werden.

Mehrere handelsübliche Zusatzmodule und -programme verwenden diese Methode, zum Beispiel auch die »Programmierhilfe« von Commodore. Verwenden Sie's doch auch! Das Kochrezept dazu steht oben in den Zeilen 50 bis 80.

Um das Ergebnis eines Tastendrucks weiter zu verarbeiten, wäre dem Computer die Abfrage zweier Speicherzellen zu langsam. Außerdem entsprechen diese Tastencodes keiner internationalen Norm, was in Verbindung mit anderen Geräten sehr lästig wäre.

Es gibt den international anerkannten ASCII-Code (American Standard Code for Information Interchange), der ursprünglich für die Zeichenübertragung von Fernschreibern erfunden wurde. Er besteht aus Dualzahlen mit einer Länge von 8 Bit oder falls Sie Dualzahlen nicht kennen (eine erste Einführung ins Dualsystem steht im Grafikkurs) aus Zahlen von 0 bis 255.

In diesen ASCII-Code wandelt der Computer nun die oben verwendeten Codezahlen der Tasten um. Die Umwandlung ist denkbar einfach.

Umrechnung des Tastatur-Codes in den ASCII-Code

Im nicht löschbaren Speicher (ROM) des Betriebssystems stehen vier Tabellen mit Zahlen.

Wenn man nun die Code-Zahl einer Taste zur Anfangsadresse der Tabellen addiert, erhält man eine Adresse, in der die ASCII-Codezahl gespeichert ist. Einfach, nicht wahr?

Probieren geht über studieren. Die Tabelle mit den unge-SHIFTeten Zeichen, also der Zahlen und Großbuchstaben, beginnt beim VC 20 ab Speicherzeile 60510, beim C 64 ab 60289.

Nehmen wir das »G«, sein Tatencode aus der Tabelle ist 19 (26).

Zu 60510 (60289) dazugezählt ergibt das 60529 (60315). Nun wollen wir mal nachschauen, was in dieser Zelle steht.

Geben Sie »direkt«, (das heißt ohne Zeilenzahl) ein:
für VC 20: PRINT PEEK (60529)
für C 64: PRINT PEEK (60315)

Das Resultat ist 71. Ein Blick auf die ASCII-Liste des Handbuches (oder in jede andere ASCII-Tabelle) zeigt uns die Richtigkeit dieser Aktion. Der ASCII-Code des Zeichens »G« ist 71.

Um alle ASCII-Codes abfragen zu können, schreiben wir wieder ein kleines Programm.

```
110 A=PEEK (203)
120 Z=60510
(120 Z=60289)
170 C=PEEK(Z+A)
```

Bitte verwenden Sie meine Zeilennummern, ich möchte nämlich später noch andere Zeilen einschieben.

Diese drei Zeilen sollten Ihnen klar sein. Zur Erinnerung: 60510 ist die Anfangsadresse der Code-Tabelle beim VC 20. Die 64er müssen also ihre eigene Adresse (60289) nehmen.

Als Ergebnis wollen wir noch den Tastaturcode (Zeile 110) und den ASCII-Code (Zeile 170) nebeneinander ausdrucken:

```
200 PRINT A;C
220 GOTO 110
```

```
LIST

5 POKE 198,10
10 POKE 631,65:POKE 632,141
20 POKE 633,156
30 POKE 634,66:POKE 635,17:POKE 636,67
99 END
READY.
RUN

READY.
A
B
 C
```

Bild 7. ▲ Die Wirkung der ASCII-Zahl 141 (SHIFT RETURN) im Beispielprogramm

```
410 PRINT "SHIFT und CLR/HOME
"
420 GET A$
430 IF A$= "" THEN 420
440 IF A$= " f-1 " THEN POKE
36879,126
450 IF A$= " f-2 " THEN POKE
36879,45
460 IF A$= " f-3 " THEN POKE
36879,25
470 IF A$= " [CTRL] " THEN POKE
36879,27
480 GOTO 420
```

Für den C 64 gelten in den Zeilen 440 bis 470 andere POKE-Adressen:
440 ..... POKE 53280,6:POKE 53281,7
450 ..... POKE 53280,5:POKE 53281,2
460 ..... POKE 53280,1:POKE 53281,1
470 ..... POKE 53280,3:POKE 53281,1

Bild 8. Programm zur Abfrage mit Gänsefüßchen

ASCII-CODE	BILDSCHIRM-CODE
0 - 31	entspricht keinem Zeichen
32 - 64	32 - 64
64 - 95	0 - 31
96 - 127 192 - 223	64 - 95
128 - 159	entspricht keinem Zeichen
160 - 191 224 - 255	96 - 127
entspricht keinem ASCII-Code	128 - 255

Bild 9. ASCII-Codes mit entsprechenden Bildschirm-Codes

Damit die beiden Zahlenbänder nicht zu schnell laufen, fügen wir noch eine Verzögerungsschleife ein.

```
210 FOR T=1 TO 200: NEXT
```

Dieser Programmablauf, der mit RUN 110 gestartet wird, reagiert jetzt auf jeden Tastendruck, links mit dem Tastencode, rechts mit ASCII. Worauf er aber nicht reagiert, sind geSHIFTete Zeichen, solche mit der Commodore-Taste »C« und die Farben (mit CTRL).

Sie wissen warum? Natürlich, denn wir fragen ja nur die Speicherzellen 203 ab und nicht zusätzlich auch 653.

Jetzt muß ich noch schnell erwähnen, daß auch der Computer diese Zellen abfragt.

Wo steht was?

Erinnern Sie sich, ich habe oben gesagt, daß im ROM Tabellen stehen:

(1) ab 60510 (60289) für normale Zeichen
(2) ab 60575 (60354) für Zeichen mit SHIFT
(3) ab 60640 (60419) für Zeichen mit C=
(4) ab 60835 (64632) für Zeichen (Farben) mit CTRL

Das bauen wir jetzt in das Programm ein:

Zeile 120 ändern wir ab. Sie fragt jetzt die Speicherzelle 653 nach den Steuertasten ab. In den Zeilen 130 bis 160 springen wir auf die vier Tabellenanfänge. Der Ausdruck in Zeile 200 schließlich wird mit der Codezahl aus 653 erweitert, nämlich B.

```
120 B=PEEK(653)
130 IF B=0 THEN Z=60510 (60289)
140 IF B=1 THEN Z=60757 (60354)
150 IF B=2 THEN Z=60640 (60419)
160 IF B=4 THEN Z=60835 (64632)
200 PRINT A;B;C
```

Die Reihenfolge der Zahlenbänder auf dem Bildschirm – entsprechend der Zeile 200 – ist jetzt von links Zeichentaste, Steuertaste, ASCII-Code.

Der ASCII-Code wird im Tastatur-Puffer abgelegt

Ehrlich gesagt, das Verfahren der Tastaturabfrage in dem vorherigen Programm ist immer noch recht kompliziert. Eine direkte Abfrage des ASCII-Codes einer Taste wäre viel besser.

Und in der Tat, der Computer bietet sie uns. Er bringt nämlich jeden ASCII-Wert in einen Speicher zur Zwischenlagerung, bis er von einem Programmschritt gebraucht wird.

Dieser Speicher heißt »Tastatur-Puffer« und liegt von Speicherzelle 631 bis einschließlich 640.

Es können also maximal zehn Werte gespeichert werden. Das erste Zeichen steht immer in 631, alle anderen werden der Reihe nach in die folgenden Zellen gebracht.

Als erstes wird das Zeichen aus 631 ausgelesen und alle anderen rücken nach.

Wenn das Programm die ASCII-Werte aus dem Tastaturpuffer auslesen kann, dann können wir das natürlich auch. Das folgende kleine Programm soll es beweisen.

```
310 PRINT CHR$(147)
330 A=PEEK(631)
```

Sie sehen, wir wollen ganz einfach in Zelle 631 des Tastaturpuffers nachschauen, welcher ASCII-Wert nach Drücken einer Taste dort steht (Zeile 330). In Zeile 340 drucken wir den Wert aus und springen dann zum PEEK-Befehl zurück.

```
340 PRINT A
350 GOTO 330
```

Wenn Sie dieses Programm mit RUN 310 laufenlassen, werden Sie merken, daß es nicht geht, ich will sagen: noch nicht geht. Den Grund dafür habe ich kurz vorher schon angedeutet.

Wo sind die Computer-Detektive? Haben Sie's gemerkt?

Nun, ich habe erklärt, daß die ASCII-Werte der gedrückten Tasten der Reihe nach im Tastaturpuffer gespeichert werden und dann, wenn ein Wert aus 631 ausgelesen wird, nachrücken. Das ist der springende Punkt: durch PEEKen lesen wir nicht aus, wir schauen nur nach!

Es gibt zwei Lösungen für dieses Problem:

1) Wir verwenden einen Befehl, der den Wert herausholt; das ist GET oder INPUT.

2) Wir gaukeln dem Computer vor, daß der Tastaturpuffer nur aus einer einzigen Speicherzelle besteht.

Die zweite Methode ist exotischer, deshalb zeige ich Sie Ihnen zuerst. Es gibt eine Speicherzelle 198. In dieser Zelle steht eine Zahl, die angibt, wieviele Zeichen im Tastaturpuffer Platz haben. Normalerweise steht da eine 10 (schauen Sie nach).

Diese Zelle kann mit kleineren Zahlen gePOKEt werden, auch mit einer 0. Die 0 löscht sozusagen den Puffer. Das machen wir jetzt in unserem Programm vor dem PEEKen:

```
320 POKE 198,0
350 GOTO 320
```

Jetzt läuft's.

Die Zeile 340 gibt uns also den ASCII-Code der gerade gedrückten Taste auf dem Bildschirm aus, zum Beispiel die Zahl 84 für das »T«, 163 für das »–« (T mit C=) und so weiter.

Unter Verwendung von GET würde unser Programm so aussehen:

```
310 PRINT CHR$(147)
320 GET A$
330 IF A$="" THEN 320
340 PRINT ASC(A$)
350 GOTO 320
```

Neu ist die Verwendung der Funktion ASC. Sie bildet den ASCII-Wert des Zeichens A$.

Für diejenigen, die es noch nicht kennen: Zeile 330 nach dem GET ist erforderlich, da der GET-Befehl nicht auf das Drücken einer Taste wartet, sondern gleich weiterläuft. Solange aber keine Taste gedrückt ist, geht die Schleife nach 320 zurück.

Mit INPUT geht's noch kürzer, nur ist die Bedienung etwas umständlicher.

```
310 PRINT CHR$(147)
320 INPUT A$
340 PRINT ASC(A$)
350 GOTO 320
```

Der Umstand liegt daran, daß INPUT im Gegensatz zu GET auf einen Tastendruck wartet, der zusätzlich mit RETURN abgeschlossen werden muß.

Nach diesen Erklärungen und Versuchen müßten Sie eigentlich in der Lage sein, Listing 1 so umzuschreiben, daß statt der Abfrage der Speicherzellen 203 und 653 der Tastaturpuffer abgefragt wird. Ich schlage Ihnen vor, daß Sie das jetzt selbst ausprobieren, sozusagen als Hausaufgabe. Die Lösung zeigt Listing 4.

Das einzige, was ich Ihnen verraten will, wissen Sie eigentlich schon, nämlich, daß Sie sich die ASCII-Werte für die f-Tasten und den Klammeraffen »@« besorgen müssen. Aber wozu haben Sie Programm 3 gleich in drei Versionen?

Ich hoffe, daß Sie nach dieser Übung einsehen, daß Sie eine vollständige Liste aller ASCII-Codezahlen und ihrer Bedeutung unbedingt brauchen. Halt, werden Sie jetzt sagen, die Liste steht ja in jedem Handbuch – sogar in dem von Commodore.

Das stimmt, nur sind die meisten Listen nicht komplett.

Erinnern Sie sich? Ich habe vorher mal gesagt, daß der

ASCII-Code die Werte von 0 bis 255 hat. Diese werden von den Commodore-Computern in nicht immer ganz der Norm entsprechender Weise für alle möglichen Zeichen und Sonderfunktionen verwendet, wie zum Beispiel die Farben, die Sonderzeichen auf den Tasten, Zeichenumschaltung und so weiter.

Auch die Funktion »Bildschirm löschen und Cursor auf HOME-Position« (das heißt die CLR/HOME-Taste) ist dabei – mit dem Codewert 147. Merken Sie was? Schauen Sie mal die jeweiligen 10er-Zeilen der Programme bisher an!

Es lohnt sich also schon, alle Code-Werte und die dazugehörigen Zeichen und Funktionen anzusehen.

Ihnen die Liste einfach abzudrucken wäre zu simpel. Sie sollen ja durch Experimentieren Ihren Computer besser kennenlernen. Ich liefere Ihnen dazu die Versuchsanordnung.

Vorher aber brauchen wir noch ein Hilfsmittel, welches uns erlaubt, aus einem ASCII-Wert das Zeichen beziehungsweise die Funktion zu ermitteln. Es ist die Umkehrung der in Programm 3 verwendeten ASC-Funktion. Sie kommt ebenfalls aus Basic und heißt CHR$(x).

Dieser Befehl liefert uns das Zeichen oder die Funktion des ASCII-Codes x.

Der Befehl PRINT CHR$(x) bringt Zeichen auf den Bildschirm. (Mit dem Befehl PRINT# a,CHR$ wird das Zeichen an ein beliebiges, mit der Nummer a bezeichnetes Peripheriegerät gebracht. Doch das will ich hier nicht weiter verfolgen.) Jetzt aber zurück zu dem Hilfsmittel.

Ergänzen Sie bitte in Programm 3 die Zeile 340 auf:

```
340 PRINT A, CHR$(A)
```

Jetzt druckt das Programm nach Start mit RUN 310 neben dem ASCII-Code auch das Zeichen, welches natürlich mit der gedrückten Taste identisch ist, auf den Bildschirm. Funktionen kann man allerdings nicht ausdrucken, sondern nur ihre Auswirkungen feststellen.

Doch nun zum Kochrezept. Der entscheidende Teil steht in Zeile 570 (Listing 5).

```
570 PRINT I;" [" CHR$(I)" ]" ;..." AAA"
```

I ist die ASCII-Codezahl, die in einer FOR-NEXT-Schleife von 0 bis 255 hochgezählt wird. Die beiden Klammern [und] stehen in Anführungszeichen, damit sie ausgedruckt werden. Zwischen ihnen soll das zum Wert I gehörige Zeichen stehen.

In den Fällen, wo der ASCII-Code nicht ein Zeichen, sondern eine Funktion bedeutet, bleibt die Klammer leer. Aber der Code wirkt sich durch die Form PRINT CHR$(I) auf die 3 A aus (die Punkte ... stellen drei Leertasten dar). Zum Beispiel erscheinen sie nach I=28 in roter Farbe, bei I=17 (Cursor Down) eine Zeile tiefer.

Das Hochzählen von I in Zeile 520 wollen wir aber ein bißchen beeinflussen, und das natürlich mit Drücken von Funktionstasten und solchen, die wir dazu verdonnern.

In Zeile 600 werden daher solche Tasten abgefragt, mit der »alten« Methode in Zelle 203. Das ist reine Willkür beziehungsweise Sentimentalität von mir, die »neue« Methode über Zelle 631 geht genauso gut.

Wenn in 203 eine 64 steht (keine Taste gedrückt), dann wartet das Programm durch Rücksprung auf die Zeile 600.

Falls wir die f1-Taste drücken, schaltet Zeile 620 den Hintergrund auf Schwarz und geht wieder in Wartestellung. Diese und die beiden anderen Farbumschaltungen mit f3 (Zeile 630) auf Weiß und mit f5 auf Hellorange (Zeile 640) sind dann sehr nützlich, wenn die Farbe der drei A gegen den Hintergrund nur schlecht oder überhaupt nicht lesbar ist.

Zeile 650 gibt uns die Möglichkeit, mit der Minus-Taste in der Liste um 1 zurückzuschalten. Zeile 530 erlaubt ein Zurückschalten über die 0 nach 25. Erst wenn irgendeine beliebige Taste gedrückt wird, springt das Programm auf Zeile 660, wo nach kurzer Zeitverzögerung der Bildschirm gelöscht (70) und I weitergezählt wird (680 und 690).

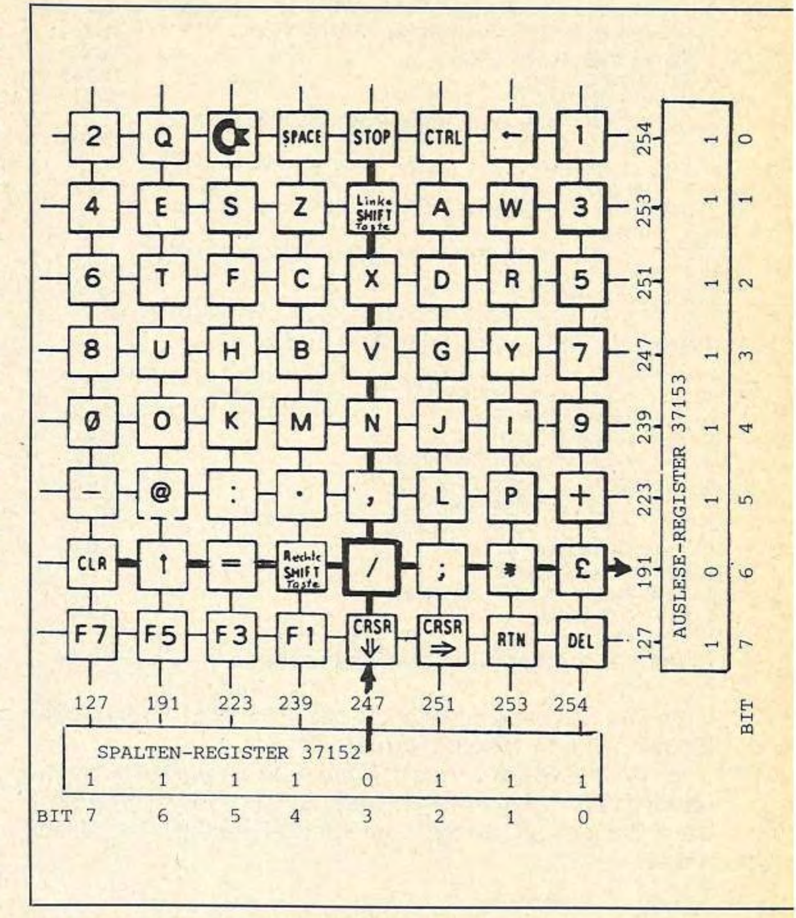


Bild 10. Die Tastenanordnung des VC 20

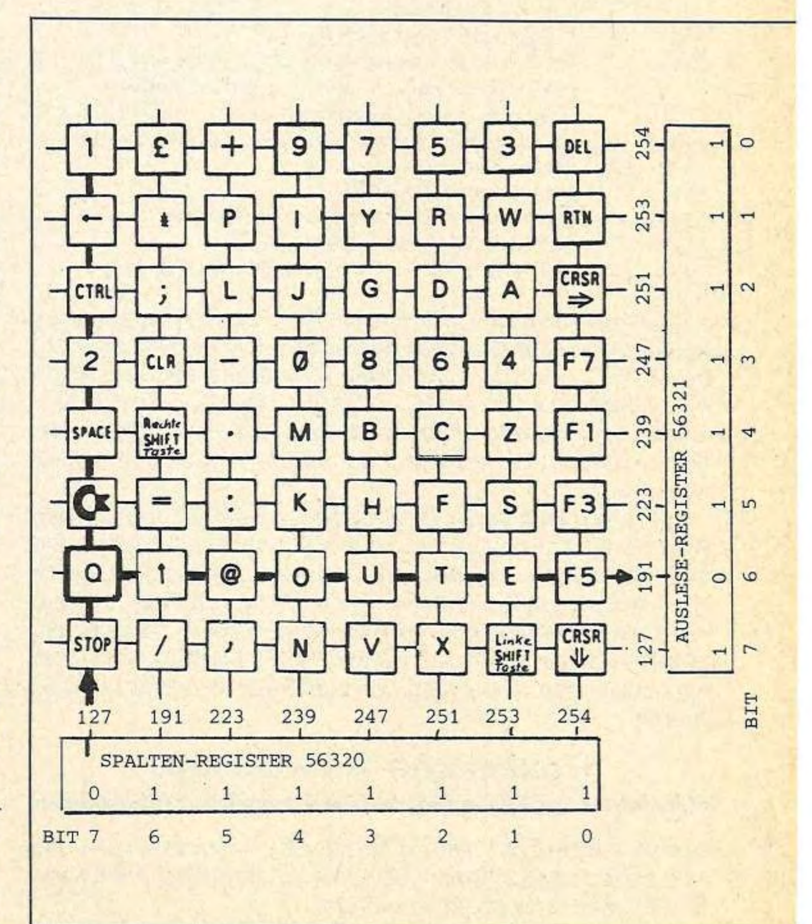


Bild 11. Die Tastenanordnung des C 64

```
5 REM*****C 64PROGRAMM 1**********          <090>
10 PRINT CHR$(147)                          <039>
20 A=PEEK(203)                              <008>
30 B=PEEK(653)                              <153>
40 IF A=4 AND B=0 THEN POKE 53280,6: POKE
   53281,2                                  <250>
50 IF A=4 AND B=1 THEN POKE 53280,5: POKE
   53281,2                                  <002>
60 IF A=5 AND B=4 THEN POKE 53280,1: POKE
   53281,1                                  <026>
70 IF A=46 AND B=0 THEN POKE 53280,3: POKE
   53281,1                                  <113>
80 GOTO 20                                  <010>
```

Listing 1. Tastenabfrage in 203 und 653

```
105 REM ****C 64 PROGRAMM 2**********       <042>
110 A=PEEK(203)                             <098>
120 B=PEEK(653)                             <243>
130 IF B=0 THEN Z=60289                     <034>
140 IF B=1 THEN Z=60354                     <131>
150 IF B=2 THEN Z=60419                     <022>
160 IF B=4 THEN Z=64632                     <191>
170 C=PEEK(Z+A)                             <251>
200 PRINT A;B;C                             <162>
210 FOR T=1 TO 200:NEXT                     <153>
220 GOTO 110                                <164>
```

Listing 2. Umrechnung des Tastencodes in ASCII-Code

So können Sie sich bequem alle 256 Werte des ASCII-Codes und ihre Wirkung anschauen.

Ich möchte Sie hier noch auf folgende Codewerte aufmerksam machen, deren Funktion Sie in diesem Programm entweder nicht sehen können oder deren Funktion den Ablauf stören:

Code	Funktionen
3	entspricht der ungeSHIFTeten STOP/RUN-Taste
8	setzt die Umschaltung (mit SHIFT und C=) auf den 2. Zeichensatz außer Betrieb (ausprobieren!)
9	hebt die Sperre wieder auf
13	entspricht der RETURN-Taste
14	schaltet den 2. Zeichensatz per Programm ein (ich empfehle, danach wieder auf den »normalen« Zeichensatz zurückzuschalten)
131	entspricht LOAD/RUN (geSHIFTete STOP/RUN-Taste)
133-140	Funktionstasten f1 bis f8
141	geSHIFTete RETURN-Taste
142	schaltet den 1. Zeichensatz ein (Umkehrung von 14)
146	REVERSE-OFF (Taste 0 mit CTRL)
160	geSHIFTete SPACE-Taste (ja, ja, das gibt es auch!)

Ich empfehle Ihnen, mit dem Listing 5 zu experimentieren. Versuchen Sie, besonders die Funktionen zu identifizieren, es ist nicht schwer. Zusätzlich sollten Sie alle sinnvollen ASCII-Werte mit den Ihnen zur Verfügung stehenden ASCII-Listen vergleichen. Verbessern und vervollständigen Sie diese Listen. Sie gehören zu Ihrem wichtigsten Handwerkszeug.

Ich habe bisher versucht, Sie zum Experimentieren mit den ASCII-Codes anzuregen. In diesem Abschnitt habe ich für Sie die vollständige Liste aller 255 Codewerte vorbereitet. Zuvor aber möchte ich Sie erst mit einem Kochrezept einstimmen, welches uns ermöglicht, auch diejenigen ASCII-Werte einzusetzen, welche durch den Befehl PRINT CHR$(.) nicht oder nur schwer darstellbar sind, wie zum Beispiel 8,8,131 und so weiter.

Dynamische Tastenabfrage

Dieses Kochrezept heißt »Dynamische Tastenabfrage« und ist mehrfach beschrieben (siehe Literaturangabe), wird aber, wie ich meine, nicht oft verwendet.

Nun bitte ich Sie, wie gewohnt am Rechner Platz zu nehmen und mir zu folgen.

Erinnern Sie sich? Der Codewert einer gedrückten Taste wird vom Betriebssystem des Rechners über eine fest eingespeicherte Tabelle in den ASCII-Codewert umgerechnet, dieser wiederum wird im »Tastaturpuffer« (Speicherplätze 631 bis 640) abgelegt. Aus diesem Puffer haben wir dann die Zahl herausgePEEKt.

Der Rechner macht genau dasselbe. Sooft wir auf eine Taste drücken, und wenn gerade kein Programm läuft, druckt er das Zeichen auf den Bildschirm oder führt die Funktion der Taste aus. Das ist der oft zitierte »Direkt-Modus«.

Wenn aber ein Programm läuft, dann bleiben die Codezahlen im Puffer so lange stehen, bis der Rechner fertig ist. Dann erst werden sie herausgeholt und verarbeitet. Das will ich Ihnen beweisen.

Tippen Sie im Direkt-Modus ein:

`FOR K=1 TO 15000:NEXT K (RETURN)`

Während diese an sich sinnlose Zeitschleife fünfzehntausendmal im Kreise rumrennt, haben Sie genügend Zeit, mehrere Tasten zu drücken, zum Beispiel die erste Buchstabenreihe (QWERTYUIOP@*↑). Natürlich sehen Sie am Bildschirm gar nichts, denn das Programm der Schleife läuft ja noch. Sobald aber die Schleife zu Ende ist, erscheinen zehn der getippten Buchstaben. Quod erat demonstrandum! Warum nur zehn Buchstaben? Nun, der Tastaturpuffer hat halt nur zehn Plätze, logisch?

Jetzt ist eine gute Gelegenheit da, nochmal die Funktion der Speicherzelle 198 auszuprobieren. In 198 kann man nämlich eine Zahl hineinPOKEn, welche die Anzahl der Zeichen im Tastaturpuffer begrenzt.

Wiederholen Sie bitte das Experiment von oben, nur soll die direkt eingegebene Zeile erweitert werden:

`FOR K=1 TO 15000:NEXT K:POKE 198,6 (RETURN)`

Und siehe da, jetzt werden nur die sechs Buchstaben Q bis Y ausgedruckt. Diese Anwendung des Tastaturpuffers nützen wir für das Kochrezept »Dynamische Tastenabfrage« aus.

Löschen Sie bitte den Bildschirm und geben Sie ein (identisch für VC 20 und C 64):

```
10 PRINT CHR$(65)
20 PRINT CHR$(165)
30 PRINT CHR$(66)CHR$(13)CHR$(67)
```

```
305 REM****C 64 PROGRAMM 3 **********       <248>
310 PRINT CHR$(147)                         <085>
320 POKE 198,0                              <230>
330 A=PEEK(631)                             <192>
340 PRINT A                                 <198>
350 GOTO 320                                <072>
405 REM****C 64 PROGRAMM 4**********        <096>
410 PRINT CHR$(147)                         <185>
420 POKE 198,0                              <074>
430 A=PEEK(631)                             <036>
440 IF A=133 THEN POKE 53280,6: POKE 53281
    ,7                                      <161>
450 IF A=137 THEN POKE 53380,5: POKE 53281
    ,2                                      <178>
```

Listing 3. Abfrage des ASCII-Codes aus dem Tastaturpuffer

```
405 REM****C 64 PROGRAMM 4**********        <096>
410 PRINT CHR$(147)                         <185>
420 POKE 198,0                              <074>
430 A=PEEK(631)                             <036>
440 IF A=133 THEN POKE 53280,6: POKE 53281
    ,7                                      <161>
450 IF A=137 THEN POKE 53380,5: POKE 53281
    ,2                                      <178>
460 IF A=134 THEN POKE 53280,1: POKE 53281
    ,1                                      <037>
470 IF A=64 THEN POKE 53280,3: POKE 53281,
    1                                       <255>
480 GOTO 420                                <210>
```

Listing 4. Tastenabfrage aus Tastaturpuffer

65 ist der Code für A, 165 für die Farbe »purple«, 66 für B, 13 für »RETURN« und 67 für C.

Bild 3 zeigt den Ausdruck auf dem Bildschirm, wenn Sie diese Zahlen LISTen und RUNen.

Zur Erklärung: Die Leerzeile zwischen A und B ist bedingt durch die PRINT-Anweisung in Zeile 20, welche nur die Farbe umschaltet. Obwohl die Codes für B und C zusammen in einer Zeile stehen, werden sie doch durch das »RETURN« (13) untereinander gesetzt. Anstelle der 13 können Sie alle möglichen anderen Steuerfunktionen setzen. Bild 4 zeigt das Resultat von 17, nämlich »CURSOR DOWN«.

Wenn Sie die 8 nehmen, können Sie den Zeichensatz nicht mehr ändern. Der Einsatz der gleichzeitig gedrückten SHIFT- und C=-Tasten funktioniert erst nach CHR$(9) wieder. Das Resultat von Bild 4 wollen wir jetzt durch POKEn der ASCII-Werte in den Tastaturpuffer versuchen.

```
5 POKE 198,5
10 POKE 631,65
20 POKE 632,156
30 POKE 633,66: POKE 634,17. POKE 635,67
99 END
```

Prinzipiell macht dieses Programm das gleiche wie das Programm in Bild 4. Trotzdem erhalten wir nach LIST und RUN ein anderes Ergebnis, nämlich das von Bild 5.

Ist das ein Fehler? Natürlich nicht. Schauen Sie her: Nach RUN laufen zuerst mal alle POKE-Befehle ab. Zeile 5 gibt an, wieviele Zeichen im Puffer stehen sollen. In Zeile 99 findet das Programm das ENDe und meldet sich mit READY. Jetzt erst wird im Tastaturpuffer nachgeschaut. Dort findet der Rechner zuerst das A, dann »purpur«, dann das B, welches sofort neben das A gesetzt wird. Das ist auch logisch, denn es fehlt ja jede Angabe, eine Zeile tiefer zu gehen. Um das zu erreichen, müssen wir in der Zeile 10 den Codewert für RETURN einschieben:

```
10 POKE 631,65: POKE 632,13
```

Vorsicht!! Sie müssen in den Zeilen 20 und 30 alle POKE-Adressen um 1 erhöhen und auch die Zahl in Zeile 5. Nehmen Sie 10, dann haben Sie Platz für Erweiterungen. So, jetzt LIST und RUN und es erscheint Bild 6 – und wir haben schon wieder ein Problem!

Aber alles im Rechner ist logisch! Nach dem A findet er den Wert für »RETURN«, also führt er den Befehl aus, auf dem der Cursor gerade steht. Der steht auf dem A. Da das kein gültiger Basic-Befehl ist, druckt der Rechner die Fehler-Meldung und zeigt READY an. Danach allerdings macht er weiter wie oben.

Und jetzt kommt die ASCII-Zahl 141 (SHIFT RETURN) voll zur Geltung. Diese Kombination nämlich setzt den Cursor an den Anfang der nächsten Zeile, ohne die Befehlsausführungsfunktion von RETURN. Ersetzen Sie also die 13 in Zeile 10 durch 141, dann läuft's (Bild 7).

Es gibt übrigens noch eine interessante ASCII-Codezahl, die in keiner Tabelle steht, nämlich 131. Sie bedeutet dasselbe wie die geSHIFTete STOP-Taste, also die Funktion »LOAD+RUN«. Wenn Sie diesen Code mit PRINT CHR$(131) ausprobieren, funktioniert er allerdings nicht. Deshalb steht er wohl auch nicht in den Tabellen. In den Tastaturpuffer gePOKEt bringt er aber seine Wirkung. Setzen Sie bitte in Zeile 30 an die Stelle von 67 die Zahl 131 und anstelle des C erscheint

```
LOAD
PRESS PLAY ON TAPE
```

So, jetzt haben wir alle Zutaten für unser Kochrezept zusammen. Löschen Sie bitte alles bisherige und tippen Sie ein:

```
10 PRINT CHR$(147)
20 FOR I=1 TO 5: PRINT I
30 FOR T=1 TO 500: NEXT T
40 NEXT T
```

Nach Löschen des Bildschirms (Zeile 10) drucken wir zum Ausschmücken die Zahlen 1 bis 5 untereinander (Zeile 20 und 40) und damit es nicht zu schnell geht, bremsen wir mit der Zeile 30.

```
50 PRINT "LIST"
```

Das ist natürlich sehr einfach, aber jetzt kommt's!

```
60 POKE 198,5
70 POKE 631,145:POKE 632,145:POKE 633,145
80 POKE 634,13
90 END
```

Nach RUN erscheinen erst die fünf Zahlen, dann wird in einer neuen Zeile das Wort LIST geschrieben. Nach END wird erst (wie immer) eine Zeile ausgelassen, dann READY gedruckt und schließlich springt der Cursor an den Anfang der Zeile darunter. Während der Cursor anfangen will, somit drei Zeilen unter dem Wort LIST zu blinken, findet der Rechner im Puffer dreimal den ASCII-Code für »CURSOR UP«. Also geht dieser auch drei Zeilen hoch und will jetzt auf dem Wort LIST blinken.

Damit wird es aber wieder nichts, dann im Puffer steht ja noch der Code für RETURN (13). Das wird ausgeführt und zwar für das LIST. Es hat dieselbe Wirkung, als ob Sie direkt LIST tippen und danach die RETURN-Taste drücken, nämlich das Programm wird ausgeLISTet.

Alle sinnvollen Basic-Befehle, die Sie in der Zeile 50 PRINTen, werden durch diese dynamische Manipulation des Cursors ausgeführt. Versuchen Sie es mit

```
50 PRINT "PRINT 16 * 35"
50 PRINT "LOAD"
50 PRINT "GOTO 10"
50 PRINT "RUN"
50 PRINT "RUN 50"
```

und falls Sie dieses kleine Programm geSAVEt haben

```
50 PRINT "SYS 64824"
(50 PRINT "SYS 64763")
```

Die Kunst ist also, mit entsprechenden Codezahlen den Cursor an diejenige Stelle des Bildschirms zu bringen, wo innerhalb eines Programms eine geeignete Anweisung gedruckt worden ist. Man kann damit getrennte Programmteile nach laden (LOAD), mit SYS-Befehlen Maschinenprogramme aufrufen, oder gar Programme durch sich selbst ändern lassen.

Die Weiterführung dieses Themas ist einen eigenen Beitrag wert, deshalb will ich hier abbrechen, denn ich bin Ihnen zum Thema »Tastenabfrage« noch einiges schuldig.

Kehren wir also wieder zur Liste der ASCII-Codes zurück. Ich habe für Sie die vollständige Liste aller 255 Codewerte vorbereitet und zwar in einer Art, die sicher einiger Erklärungen bedarf.

ASCII-Codes, die von Commodore verschwiegen werden

Ich habe nämlich noch ein paar zusätzliche Überraschungen parat, auf die man nur durch Zufall kommt oder durch Studium des Betriebssystems oder aber, wenn man den Aufsatz von G. Urbanczyk in Computer persönlich vom 19.10.1983, Seite 76, gelesen hat.

Tippen Sie bitte Listing 3 ein, nämlich eine der drei Versionen zur Tastaturpuffer-Abfrage. (Ich verwende unten die GET-Version).

```
310 PRINT CHR$(147)
320 GET A$
330 IF A$= "  " THEN 320
340 PRINT ASC(A$)
350 GOTO 320
```

Auf beiden, VC 20 und C 64 erhalten Sie nach RUN 310

```
505   REM *** Programm 5 ******************
506   REM DER VOLLSTÄNDIGE ASCII-CODE *
507   REM **********************************
510   PRINT CHR$(147)
520   FOR I=0 TO 255
530   IF I<0 THEN I=255
540   PRINT:PRINT:PRINT:PRINT:PRINT
550   PRINT"-----------"
560   PRINT
570   PRINT I;"[" CHR$(I);"]";"... AAA"
580   PRINT
590   PRINT "-----------"
600   Z=PEEK(203)

610   IF Z=64 THEN 600
620   IF Z=4 THEN POKE 53280,3: POKE 53281,0: GOTO 600
630   IF Z=5 THEN POKE 53280,3: POKE 53281,1:GOTO 600
640   IF Z=6 THEN POKE 53280,3: POKE 53281,10: GOTO 600
650   IF Z=43 THEN I=I-2

660   FOR T=1 TO 100:NEXT T
670   PRINT CHR$(147)
680   NEXT I
690   GOTO 520
```

Listing 5. Der vollständige ASCII-Code

und Drücken der RETURN-Taste (natürlich) den ASCII-Code 82. Wenn Sie zuerst die CTRL-Taste drücken und halten und dann erst das R drücken, dürfte eigentlich nichts passieren, denn die CTRL-Taste gilt ja angeblich nur für die Farben. Ja, denkste! Wir erhalten nämlich die Zahl 18. Ein Blick in die ASCII-Tabelle zeigt uns für 18 die Funktion »REVERSE-ON«.

Versuchen Sie dasselbe mit CTRL und der ←-Taste. Wir erhalten die 6, und nicht 95, wie es eigentlich sein sollte.

Für den VC20 ist das alles. Aber immerhin, wir haben sozusagen noch zwei zusätzliche Funktionstasten gefunden.

Beim C 64 aber geht es erst richtig los:

Der Versuch wird Ihnen zeigen, daß alle Buchstaben, von A bis Z, zusammen mit CTRL gedrückt, einen anderen ASCII-Wert, nämlich 1 bis 26, ergeben, als allein gedrückt.

Des weiteren biete ich Ihnen noch:

CTRL - ↑ = 30
CTRL - = = 31
CTRL - $ = 28
CTRL - : = 27
CTRL - ; = 29

Das heißt aber, daß einige ASCII-Codezahlen zwei Bedeutungen haben. Oder umgekehrt, zwei verschiedene Tasten (kombiniert) haben denselben ASCII-Code.

Schwierigkeiten dadurch, daß einige ASCII-Werte zwei Bedeutungen haben, gibt es deswegen nicht, weil die Kombination mit CTRL nicht PRINT-bar ist (PRINT CHR$(19) schickt immer den Cursor »home«, mit dem »S« passiert gar nichts).

Andersherum kann es allerdings vorkommen, daß eine Tastenabfrage, zum Beispiel

```
GET A$:IF A$ = CHR$(19) THEN ......
```

sowohl auf die Taste »HOME« als auch auf »CTRL-S« reagiert. Da ist sicher etwas Vorsicht angebracht. Aber ein Blick in meine ASCII-Tabelle zeigt Ihnen ja die Doppeldeutigkeiten.

An dieser Stelle erwarte ich eigentlich einige Einsprüche, wie: »Wozu das alles, die acht Funktionstasten, oder gar erweitert auf 32, reichen doch völlig aus!« Für den Hausbeziehungsweise Spielgebrauch ist das sicher richtig. Aber bei professioneller Software, welche benutzerfreundlich aufgebaut ist, kann es oft gar nicht genügend Funktionstasten - besonders solche, die eine optische Buchstabenbeziehung zu der Abfrage haben sollen - geben. Wenn in einem Programm gefragt wird, ob Sie »LOADen« oder »SAVEn« wollen, ist CTRL-L oder CTRL-S halt klarer als f-1 oder f-3.

Ich finde es schade, daß diese großartige Möglichkeit nur auf dem C64 gegeben ist, der doch professioneller ist als der VC 20.

Die vollständige ASCII-Tabelle

So, jetzt können Sie meine ASCII-Tabelle erst richtig interpretieren (Tabelle 2).

Leere Kästchen haben keine Bedeutung für die betreffende Codezahl.

Jeweils zwei Zeichen nebeneinander mit derselben Codezahl stellen die beiden Zeichensätze dar, in die mit C=SHIFT (Commodore-Taste) umgeschaltet werden kann. Wo nur ein Zeichen steht, ist es in beiden Zeichensätzen identisch.

Die Funktionen der Codezahlen 129 und 149 bis 155 gelten nur für den C64. Interessant ist übrigens, daß die 4. und 7. Spalte identisch ist, ebenso die 6. und 8. Spalte (außer dem Zeichen für 255).

Ich möchte jetzt gern die Szene wechseln, ohne aber den ASCII-Code aus den Augen zu verlieren. Wir haben den ASCII-Code bisher verwendet, um Tasten abzufragen oder Funktionen auszuführen. PRINT CHR$(66) druckt zum Beispiel den Buchstaben B auf den Bildschirm.

Welche Methoden kennen Sie noch, mit denen das gleiche erzielt werden kann?

Die erste, die jeder aus dem Handbuch lernt, ist

```
PRINT " B" .
```

Die komplizierteste ist:

```
POKE 7680,2: POKE 38400,7
(POKE 1024,2: POKE 55296,7)
```

Diese beiden Vorgehensweisen wollen wir uns näher anschauen und prüfen, ob wir sie in Analogie zu dem ASCII-Code für Tastenabfragen einsetzen können.

Der Gänsefüßchen-Modus

Es ist sicher viel bequemer, längere Buchstabenreihen oder gar Texte zwischen Gänsefüßchen gestellt einzutippen als eine Serie von CHR$-Werten, ganz abgesehen vom erforderlichen Speicherplatz.

Nicht ganz so bequem ist der Gänsefüßchen-Modus bei Steuerzeichen, wie zum Beispiel Cursor-Bewegungen, besonders, wenn man diese herbeiführen will, aber statt dessen die reversen Darstellungen auf dem Bildschirm erzeugt.

Geben Sie es zu. Sie haben deswegen auch schon herzhaft geflucht. Auch jeder Redakteur bittet um Listings mit CHR$-Darstellung anstelle der reversen Zeichen, die bei der Druckwiedergabe oft zu Schwierigkeiten führen.

Jetzt wissen Sie, warum ich bei meinen Progrämmchen immer PRINT CHR$(147) statt PRINT"   " verwende.

Genauso austauschbar wie bei PRINT ist der ASCII-Code mit dem Gänsefüßchen-Modus bei der Tastenabfrage.

Statt:

```
10 GET A$
20 IF A$<>CHR$(65) THEN 10
30 PRINT CHR$(88)
```

können wir schreiben:

```
10 GET A$
20 IF A$<>" A" THEN 10
30 PRINT " X"
```

Beide Programme sind gleichwertig. Nach RUN rührt sich gar nichts. Erst, wenn die A-Taste gedrückt wird (Zeile 20), druckt Zeile 30 den Buchstaben X.

In Zeile 20 können wir natürlich statt des A jeden beliebigen Buchstaben, Zahl oder Zeichen nehmen.

LISTen Sie einfach die 2. Version der drei Zeilen, fahren mit dem Cursor auf das A und verändern Sie es Ihren Wünschen entsprechend. Wie ich Sie einschätze, machen Sie das sicher auch mit den Funktionstasten.

Nein? Dann machen Sie es mal. Sie haben nämlich Pech, so geht es nicht. Aber es geht, wenn Sie sich mit Absicht in die Lage begeben, die wie vorhin beschrieben, Flüche auslöst. Fahren Sie mit dem Cursor auf das 1. Gänsefüßchen, tippen Sie es noch einmal ein und drücken Sie dann eine Funktionstaste. Siehe da, es erscheint ein reverses Zeichen. Mit RETURN wird es »fixiert«, nach RUN wird das X erst mit der verwendeten Funktionstaste ausgelöst.

Der Trick besteht also darin, durch eine ungerade Anzahl von Gänsefüßchen-Eingaben diesen Modus herbeizuführen. Es geht ebenso durch Drücken der INSERT(INST)-Taste, allerdings nur für soviele Zeichen, wie oft sie gedrückt worden ist.

Im Gänsefüßchen-Modus erscheinen alle Steuer- und Funktionstasten in reverser Darstellung

Sie haben oben ein reverses Zeichen für die Funktionstasten erhalten. Die Zeichen für die Farben und Cursorbewegungen, also alle »gängigen« Funktionen, kennen Sie inzwischen sicher schon. Aber alle Steuer- und Funktionstasten?

Es gibt zwei Möglichkeiten, diese Zeichen zu finden: Die erste Methode verwendet entweder ganz primitiv im Direkt-Modus den Befehl: PRINT " mit nachfolgendem Drücken der Steuer- oder Funktionstaste oder sehr elegant den Dreizeiler

```
10 GET A$: IF A$="   " THEN 10
20 PRINT CHR$(34) A$ CHR$(34) ASC(A$)
30 GOTO 10
```

Auch hier ist ein kleiner Pfiff drin. In Zeile 20 wird zuerst ein Gänsefüßchen (34) gedrückt, wodurch wir in dem danach benannten Modus sind. Das nachfolgende A$ erscheint im Fall einer Steuertaste als reverses Zeichen, nach dem abschließenden 2. Gänsefüßchen gibt uns die ASC-Funktion noch den ASCII-Code der gedrückten Taste.

Mit Gänsefüßchen statt CHR$ hätten wir lediglich die beiden Zeichen A und $ auf den Bildschirm gedruckt. Mit dieser Methode erhalten Sie zum Beispiel für:

ASCII-Code 17 = ▮
Cursor Down . . . .
ASCII-Code 147 = ▮
CLR . . . .

Die zweite Methode ist viel einfacher. Schauen Sie in meine ASCII-Tabelle (Tabelle 2). Sie ist in Spalten zu je 32 Zeichen angeordnet. Die Steuerzeichen und die Farben stehen alle in Spalte 1 und 5.

Da finden Sie zum Beispiel über der Codezahl 17 die Funktion »Cursor-Down«. Wenn Sie jetzt in die 3. Spalte waagrecht rübergehen – also den Wert um 64 erhöhen – steht da das ▮. Oder: In Spalte 5 ist der Taste CLR die Codezahl 147 zugeordnet. Zwei Spalten weiter (64 höher) steht das ▮.

Wenn Sie die Ergebnisse der ersten Methode oben mit den durch Spaltenhüpfen gefundenen Zeichen vergleichen, sehen Sie, daß es dieselben Zeichen sind, halt nur reversiert. Machen wir die Probe:

Mit Methode 1 erhalten wir für »Rot« das reverse Pfund-Zeichen ▮. In der ASCII-Tabelle finden wir »Rot« unter 28. Zwei Spalten weiter, unter 28+64=92, steht dasselbe Zeichen. Das gilt auch für alle CTRL-Kombinationen, nicht nur für die der Farben.

```
5 REM************************************
6 REM***** SPIEL MIT FINKELN **********
7 REM************************************
10 PRINT CHR$(147)
20 F=0
30 R=65
40 FOR T=1 TO 600:NEXT T
50 A=INT(RND(0)*7)+65
60 IF R>71 THEN 400
70 PRINT CHR$(A);
80 FOR T=1 TO 1000:NEXT T
100 GET A$
110 IF A$<>"▮" THEN 300
120 PRINT"▮";
200 IF A=R THEN F=F+1
210 GOTO 50
300 IF A<>R THEN F=F+1
310 R=R+1
320 GOTO 50
400 PRINT"▮▮▮DAS SPIEL IST ZU ENDE";"▮
"F"FEHLER"
```

Listing 6. Programmtext zum »Spiel mit Finkeln« für C64

Bei beiden Computern entspricht dem CTRL-▮ das ▮, beim C 64 erzeugt CTRL-▮ ein ▮. Alle Kombinationen der Buchstaben mit CTRL erzeugen diese Buchstaben in reverser Darstellung.

Um das in einer kleinen praktischen Anwendung zu verdeutlichen, schlage ich vor, dieselbe Aufgabenstellung, die mit Tastencode-Abfrage und mit Tastaturpuffer-Abfrage gelöst wurde, noch einmal zu verwenden, jetzt aber die Gänsefüßchen-Methode einzusetzen.

Um beim Eintippen des Programms in Bild 8 sicherzustellen, daß alles klappt, habe ich statt der reversen Zeichen die Tasten angegeben beziehungsweise umrahmt, die nach dem ersten Gänsefüßchen gedrückt werden müssen.

Das Programm schaltet, wie die beiden anderen Versionen auch, die Bildschirm-Farben mit f-1, f-2, f-3 und @ um.

Ein letztes Problem bleibt uns noch. Wie schaffen wir es, daß wir im Gänsefüßchen-Modus auch Funktionen einsetzen können, die entweder keine eigene Taste haben (zum Beispiel 14 = Text, 8 = Lock) oder die beim Eintippen sofort die Funktion auslösen (zum Beispiel 13 = RETURN, 20 = DELete)?

Hier müssen wir eine Methode anwenden, die meine Kinder und ich »finkeln« getauft haben und zwar deswegen, weil wir sie zum ersten und einzigen Mal vom Commodore-Software-Spezialisten Andy Finkel im amerikanischen Handbuch gefunden haben.

Sein Trick besteht darin, daß er in einer ASCII-Tabelle das entsprechende Zeichen für die Funktion heraussucht und es in mehreren Schritten an seinen vorgesehenen Platz bringt.

Ich will Ihnen zeigen, was ich damit meine:

```
3 REM****************
4 REM**CODE-WANDLER**
5 REM****************
10 PRINT CHR$(147)
20 INPUT "▮ASCII-CODE";ACII
30 IF ACII=0 THEN END
40 IF ACII AND 128 THEN BILD=ACII AND 127 OR 64:GOTO 80
50 IF NOT ACII AND 64 THEN BILD=ACII:GOTO 80
60 IF ACII AND 32 THEN BILD=ACII AND 95:GOTO 80
70 BILD=ACII AND 63
80 PRINT TAB(5) "BILDSCH.CODE:"BILD
90 GOTO 20
```

Listing 7. Ein ASCII-Code Wandler

```
4 REM **********************
5 REM    VIELFACHTASTEN
6 REM       ABFRAGE
7 REM **********************
10 PRINT CHR$(147)
20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29
100 DATA 169,254,141,0
101 DATA 220,173,1,220
102 DATA 141,255,29,96
105 REM ---------VC20-----------
106 REM    100 DATA169,254,141,32
107 REM    101 DATA145,173,33,145
108 REM    102 DATA141,255,29,96
109 REM ---------VC20-ENDE------
110 FOR K=1 TO 12
120 READ X
130 POKE 7660+K,X
140 NEXT K
200 POKE 7662,127
210 SYS 7661
220 PRINT " "PEEK(7679);
300 POKE 7662,191
310 SYS 7661
320 PRINT PEEK(7679);
400 POKE 7662,223
410 SYS 7661
420 PRINT PEEK(7679);
500 POKE 7662,239
510 SYS 7661
511 REM  -------VC20-------------
512 REM  520 PRINT PEEK(7679)
513 REM  550 GET A$
514 REM  560 IF A$<>""THEN200
515 REM  -------VC20-ENDE--------
520 PRINT PEEK(7679);
600 POKE 7662,247
610 SYS 7661
620 PRINT PEEK(7679);
700 POKE 7662,251
710 SYS 7661
720 PRINT PEEK(7679);
800 POKE 7662,253
810 SYS 7661
820 PRINT PEEK(7679);
900 POKE 7662,254
910 SYS 7661
920 PRINT PEEK(7679)
940 REM --------VC20--------------
950 REM  GET A$
960 REM  IF A$<>"" THEN 600
970 REM -------VC20-ENDE--------
1000 GOTO 200

READY.
```

Listing 8. Programm zum Abfragen und Anzeigen aller Tasten

Bitte, versuchen Sie mit der Gänsefüßchen-Methode die DELete-Taste in eine PRINT-Anweisung zu bringen
10 PRINT " INST/DEL "
Sie werden es nicht schaffen, da die DEL-Taste, statt ein reverses Zeichen zu drucken, ihrer Funktion nachgeht und das vorherige Zeichen löscht.
Jetzt »finkeln« wir:
1. Schritt:
10 PRINT " "
(mit RETURN abschließen)

```
0 REM****************
1 REM** SPIEL FUER **
2 REM** 4 PERSONEN **
3 REM****************
4 REM
5 REM
6 REM****************
7 REM*** EINGABE ****
8 REM**MASCHINENCODE*
9 REM****************
10 PRINT CHR$(147)
20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29
VC 20: 100 DATA 169,0,141,32,145,173,33,145,141,255,29,96
C 64: 100 DATA 169,0,141,0,220,173,1,220,141,255,29,96
110 FOR K=1 TO 12
120 READ X
130 POKE 7660+K,X
140 NEXT K
199 REM**************
200 REM*ANWEISUNGEN**
205 REM**************
210 PRINT CHR$(147)
215 POKE830,0
VC 20:
220 PRINT"SPIELER 1:(W)TASTE
230 PRINT"SPIELER 2:(Z)TASTE
240 PRINT"SPIELER 3:(DEL)TASTE
250 PRINT"SPIELER 4:(CRSR←)TASTE
C 64:
220 PRINT"SPIELER 1 DRUECKT '←'
230 PRINT"SPIELER 2 DRUECKT 'COMMODORE'
240 PRINT"SPIELER 3 DRUECKT 'INST/DEL'
250 PRINT"SPIELER 4 DRUECKT 'CRSR←'
260 PRINT TAB(125)"SPACE"
270 GET A$:IF A$=""THEN 270
280 PRINT CHR$(147)
300 POKE 36879,8
310 PRINT"WER SIEHT ALS ERSTER
320 PRINT"DEN BUCHSTABEN
330 BU = INT(RND(0)*10)+1
VC 20:
340 POKE 7808,BU
350 POKE 38528,7
360 PRINT"UND DRUECKT SEINE     TASTE ?
C 64:
340 POKE 1224+17,BU
350 POKE 55496+17,7
360 PRINT"UND DRUECKT SEINE TASTE ?
370 PRINT TAB(165)"SPACE"
380 GET A$:IF A$="" THEN 380
390 PRINT CHR$(147)
400 FOR T=1 TO 800:NEXT T
490 REM**************
495 REM* BUCHSTABEN *
500 REM** WUERFELN **
510 REM**************
520 VB=INT(RND(0)*10)+1
530 IF VB=BU THEN POKE 830,250
VC 20:
540 C=INT(RND(0)*505)
550 POKE 7680+C,VB
560 POKE 38400+C,1
C 64:
540 C=INT(RND(1)*501)*2
550 POKE 1024+C,VB
560 POKE 55296+C,1
590 REM**************
595 REM** ABFRAGE ***
600 REM* DER TASTEN *
605 REM**************
610 FOR Z=1 TO 15
VC 20:
620 POKE 7662,253
630 SYS 7661
640 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 810
650 POKE 7662,239
660 SYS 7661
670 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 820
680 POKE 7662,254
690 SYS 7661
700 IF PEEK(7679)=127 AND PEEK(830)=250 THEN 830
710 POKE 7662,251
720 SYS 7661
730 IF PEEK(7679)=127 AND PEEK(830)=250 THEN 840
740 NEXT Z
```

Listing 9. Ein Tastenspiel für vier Personen

2. Schritt:
Mit dem Cursor auf die Leerstelle zwischen den Gänsefüßen fahren.
3. Schritt:
Aus der ASCII-Tabelle das Zeichen der DEL-Taste holen (T).
4. Schritt:
Die reverse Darstellung mit CTRL-REV.ON einschalten (der Cursor bleibt auf seiner Stelle) und das T drücken, mit RETURN abschließen. Jetzt steht das Zeichen drin und das Programm läuft.

Um Ihnen den Schritt 3 für alle widerborstigen Funktionen zu erleichtern, habe ich sie alle in der Tabelle 3 zusammengefaßt.

Da ich hoffe, daß Sie in Zukunft fleißig finkeln werden, muß ich Sie noch über einen lästigen Nebeneffekt aufklären, der bei ein paar Finkeleien auftritt. Einige der Funktionen, nämlich RETURN, DELete (schon wieder) und das SHIFT-RETURN wirken nicht nur im Programmablauf wie vorgesehen, sondern auch beim LISTen, was lästig sein kann. Allerdings ergeben sich dadurch auch ungeahnte Möglichkeiten – siehe Artikel »Synthetische Steuerzeichen«. Das geSHIFTete RETURN (ASCII-Code 14) ist sehr nützlich bei Platz- und Speichermangel. Sie können nämlich mit " ▮ " in einer langen Programmzeile den Cursor mit nur drei Zeichen auf den Anfang der nächsten Zeile bringen, mit CHR$(141) bräuchten Sie schon neun Zeichen, mit SPC(...) müssen Sie sehr genau die Cursorposition berechnen, mit einer entsprechenden Anzahl von »Cursor-Rechts«-Zeichen geht es auch nur mühsam.

Also, nützlich ist SHIFT-RETURN durchaus!

Nur: Beim LISTen wird es auch ausgeführt und die Zeile, in der es steht, sieht recht blöd aus. Zusätzlich kann eine derart geLISTete Zeile nicht mehr geändert werden, sondern muß bei Verbesserungen völlig neu geschrieben und gefinkelt werden. Alles Gute hat seinen Preis!

Soviel sei zur Methode gesagt. Jetzt wollen wir zur Erholung und zur Übung ein kleines Spiel programmieren, in dem wir (fast) alles Gelernte auch anwenden.

Eine kleine Seltenheit ist bemerkenswert: Das Programm ist für VC 20 und C 64 identisch!

Die Spielaufgabe soll darin bestehen, die ersten sieben Buchstaben des Alphabets möglichst in der richtigen Reihenfolge auf den Bildschirm zu bringen.

Klingt einfach, aber die Buchstaben sollen in zufälliger Reihenfolge auftauchen. Zusätzlich hat der Spieler, falls der Buchstabe nicht der Reihenfolge entspricht, lediglich die Möglichkeit, ihn mit der DEL-Taste zurückzuweisen, wenn er schnell genug ist. Das Programm zählt die Felder und zeigt am Schluß das Ergebnis an.

Wir brauchen dazu:
- Einen Zufalls-Buchstaben-Erzeuger von A bis G (ASCII-Code 65 bis 71)
- einen Buchstaben-Drucker
- einen Buchstaben-Reihenfolgezähler
- eine Möglichkeit, die DEL-Taste zu drücken und damit den gedruckten Buchstaben rückgängig zu machen
- einen Fehlerzähler
- eine Prüfung, ob der letzte Buchstabe (71) erreicht ist.

Normalerweise müßte ich jetzt ein Flußdiagramm zeichnen und »strukturiert« vorgehen, so wie die ausgezeichnete Serie in diesem Heft lehrt. Man möge mir aber verzeihen, daß ich aus Erklärungsgründen in einzelnen Schritten vorgehe, welche uns erlauben, jederzeit Zwischenresultate mit Probeläufen zu überprüfen (Listing 6).
Auf geht's!

Den Buchstaben-Erzeuger und -drucker erhalten wir durch Zeile 50, welche für eine Variable A zufällige ASCII-Codes zwischen 65 und 71 erzeugt, sowie durch Zeile 70, die das Zeichen für den ASCII-Code ausdruckt.

```
50 A=INT(RND(0)*7)+65
70 PRINT CHR$(A);
```

Für weniger Versierte sei gesagt, daß RND(0) eine Zufallszahl zwischen 0 und 0,99 erzeugt, mit 7 multipliziert gibt das eine Zahl zwischen 0 bis 6,93. Die Funktion INT macht daraus eine ganze Zahl zwischen 0 und 6, mit 65 addiert letztlich eine Zahl zwischen 65 und 71 = ASCII-Werte der Buchstaben A bis G.

Den Ausdruck der Buchstaben nebeneinander erreichen wir durch das Semikolon in Zeile 70, die laufende Wiederholung durch einen Rücksprung in Zeile 320.

```
320 GOTO 50
```

Damit es nicht zu schnell geht, verzögern wir das Ganze mit einer Warteschleife in Zeile 80.

```
80 FOR T=1 TO 1000:NEXT T
```

Probieren Sie es mit RUN aus. Zeile 80 übrigens erlaubt Ihnen später den Schwierigkeitsgrad zu verändern.

Die geforderte richtige Reihenfolge der Buchstaben A, ich nenne sie hier R, setzen wir am Anfang auf 65 und erhöhen sie schrittweise um 1.

```
30 R = 65
310 R = R+1
```

In Zeile 60 prüfen wir, ob die Endzahl 71 für das G überschritten ist. Wenn ja, springen wir auf Zeile 400, mit der wir das Spielende anzeigen.

C 64

```
620 POKE 7662,127
630 SYS 7661
640 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 810
650 IF PEEK(7679)=223 AND PEEK(830)=250 THEN 820
660 IF PEEK(7679)=221 AND PEEK(830)=250 THEN 850
670 POKE 7662,254
690 IF PEEK(7679)=254 AND PEEK(830)=250 THEN 830
700 IF PEEK(7679)=251 AND PEEK(830)=250 THEN 840
710 IF PEEK(7679)=250 AND PEEK(830)=250 THEN 850
720 NEXT Z
750 GOTO 500: REM (NEUER BUCHSTABE)
799 REM**************
800 REM** ERGEBNIS **
805 REM**************
810 PRINT"▮▮▮▮SPIELER 1 IST SIEGER":GOTO 860
820 PRINT"▮▮▮▮SPIELER 2 IST SIEGER":GOTO 860
830 PRINT"▮▮▮▮SPIELER 3 IST SIEGER":GOTO860
840 PRINT"▮▮▮▮SPIELER 4 IST SIEGER":GOTO 860
850 PRINT"▮▮▮ZWEI SPIELER WAREN GLEICHZEITIG...
860 PRINT"▮▮▮NOCH EINMAL ?  (J/N)
870 GET A$:IF A$="J" THEN 200
880 IF A$="N" THEN END
890 GOTO 870
READY.
```

**Listing 9.
Ein Tastenspiel
für vier Personen (Schluß)**

64er-online.de

TASTE	VC-20		C-64	
	203	653	203	653
nichts	64	0	64	0
f-1	39	0	4	0
f-3	47	0	5	0
f-5	55	0	6	0
f-7	63	0	3	0
A	17	0	10	0
B	35	0	28	0
C	34	0	20	0
D	18	0	18	0
E	49	0	14	0
F	42	0	21	0
G	19	0	26	0
H	43	0	29	0
I	12	0	33	0
J	20	0	34	0
K	44	0	37	0
L	21	0	42	0
M	36	0	36	0
N	28	0	39	0
O	52	0	38	0
P	13	0	41	0
Q	48	0	62	0
R	10	0	17	0
S	41	0	13	0
T	50	0	22	0
U	51	0	30	0
V	27	0	31	0
W	9	0	9	0
X	26	0	23	0
Y	11	0	25	0
Z	33	0	12	0
1	0	0	56	0
2	56	0	59	0
3	1	0	8	0
4	57	0	11	0
5	2	0	16	0
6	58	0	19	0
7	3	0	24	0
8	59	0	27	0
9	4	0	32	0
0	60	0	35	0
+	5	0	40	0
-	61	0	43	0
*	14	0	49	0
/	30	0	55	0
=	46	0	53	0
↑	54	0	54	0
←	8	0	57	0
.	37	0	44	0
:	45	0	45	0
,	29	0	47	0
;	22	0	50	0
£	6	0	48	0
@	53	0	46	0
CRSR←	23	0	2	0
CRSR↑	31	0	7	0
DEL	7	0	0	0
HOME	62	0	51	0
STOP	24	0	63	0
RETURN	15	0	1	0
SPACE	32	0	60	0
SHIFT	64	1	64	1
C=	64	2	64	2
CTRL	64	4	64	4
SHIFT u. C=	64	3	64	3
SHIFT u.CTRL	64	5	64	5
C= u. CTRL	64	6	64	6
SHIFT u. C= u.CTRL	64	7	64	7

Tabelle 1. Tabelle des Tastencodes in den Registern 203 und 653. Eine komplette Tabelle, in der auch die Codes für Grafikzeichen enthalten sind, finden Sie im Sonderheft 8/85 (Assembler)

```
60 IF R > 71 THEN 400
400 PRINT "███
SPIELENDE"
```

Bitte RUNnen Sie das Fragment wieder zur Probe.

Jetzt kommt die Beeinflussung der Reihenfolge mit der DEL-Taste. Wie gelernt fragen wir diese Taste mit einer GET-Schleife ab (Zeilen 100,110), ihre Lösch-Wirkung erreichen wir in Zeile 120 durch einen PRINT-Befehl (mit Semikolon!). Nach Drücken der DEL-Taste darf der Reihenfolge-Zähler der Zeile 310 natürlich nicht wirken, deshalb springen wir schon vorher aus der Zeile 210 zurück.

```
100 GET A$
110 IF A$<>" █ ". THEN 300
120 PRINT " █ ";
210 GOTO 50
```

Sie sehen oben, daß ich für die Abfrage der DEL-Taste die Finkel-Methode vorschlage. Die anderen Methoden gehen natürlich auch.

Nach RUN springt das Programm auf die noch nicht existierende Zeile 300 (was prompt zur Fehlermeldung führt), es sei denn, Sie drücken rechtzeitig die DEL-Taste.

In der Zeile 300 wollen wir prüfen, ob ein Fehler gemacht wurde, das heißt, ob A mit der Reihenfolge R übereinstimmt. Im Fehlerfall wird die Fehlerzahl F um 1 erhöht. Vorher aber muß F auf 0 gesetzt werden.

```
20 F=0
300 IF A<>R THEN F=F+1
```

Sie können jetzt schon das Spiel üben. Aber es fehlen noch ein paar Feinheiten.

```
10 PRINT CHR$(147)
410 PRINT F " Fehler"
```

Zeile 10 ist klar, Zeile 410 druckt am Spielende die Fehlerzahl F aus.

Aber es gibt noch einen Fehler des Spielers, nämlich wenn er aus Versehen einen richtigen Buchstaben zurückweist. Deshalb fragen wir nach erfolgtem Drücken der DEL-Taste in den Zeilen 100 bis 120 nach, ob der Buchstabe tatsächlich falsch war. Wenn nicht, wird die Fehlerzahl F um 1 erhöht.

```
200 IF A=R THEN F=F+1
```

Damit uns nach RUN der erste Buchstabe nicht überrascht, verzögern wir sein Erscheinen mit

```
40 FOR T=1 TO 600: NEXT T
```

Zum Finkeln-Üben arrangieren wir die Anzeige des Spielendes und der Fehler etwas um. Alle Anweisungen sollen in nur einer Zeile stehen. Löschen Sie bitte die Zeile 410. In Zeile 400 wird gefinkelt und zwar mit dem Zeichen für SHIFT RETURN, welches laut Tabelle 2 mit SHIFT M erzeugt wird.

```
400 PRINT "████ DAS SPIEL IST ZU ENDE";"████"
  F " FEHLER".
```

Bei LIST und bei Ausdruck mit einem Drucker sehen die gefinkelten Zeilen 110, 120 und 400 natürlich kurios aus und wie gesagt, sie lassen sich bei einem Tippfehler nicht korrigieren, sondern müssen neu geschrieben werden.

Der Bildschirm-Code

Der Vollständigkeit halber will ich noch die letzte der vorher genannten drei Methoden erwähnen, mit denen man ein Zeichen auf den Bildschirm bringt, insbesondere, weil der dabei verwendete Bildschirm-Code (auch Video-Code genannt) oft zu Verwechslungen mit dem ASCII-Code führt.

Auf Anhieb ist es auch nicht einzusehen, warum Commodore einen anderen Code verwendet, wenn ein Zeichen direkt auf den Bildschirm – oder genauer gesagt, in den Bildschirm-Speicher – gePOKEt werden soll.

Der Grund dafür liegt darin, daß dem ASCII-Code nicht nur Zeichen zugeordnet sind, sondern auch Farben und Funktio-

KOMBINATIONEN		1
VC 20	C 64	
		000
	CTRL — A	001
	CTRL — B	002
	CTRL — C	STOP 003
	CTRL — D	004
	CTRL — E	WHT 005
CTRL — ←	CTRL — F CTRL — ←	006
	CTRL — G	007
	CTRL — H	LOCK 008
	CTRL — I	UN LOCK 009
	CTRL — J	010
	CTRL — K	011
	CTRL — L	012
	CTRL — M	CAR RET 013
	CTRL — N	TEXT 014
	CTRL — O	015
	CTRL — P	016
	CTRL — Q	CRSR DOWN 017
CTRL — R	CTRL — R	RVS ON 018
	CTRL — S	CRSR HOME 019
	CTRL — T	DEL 020
	CTRL — U	021
	CTRL — V	022
	CTRL — W	023
	CTRL — X	024
	CTRL — Y	025
	CTRL — Z	026
	CTRL — :	027
	CTRL — £	RED 028
	CTRL —]	CRSR RT 029
	CTRL — ↑	GRN 030
	CTRL — =	BLU 031

64er-online.de



Tabelle 2. ASCII-Code

nen. Außerdem sind im ASCII-Code die reversen Zeichen nicht enthalten, sondern müssen – wie Sie ja inzwischen wissen – jeweils umgeschaltet werden. Das alles ist für ein Betriebssystem viel zu kompliziert.

Es ist viel einfacher, im Festspeicher (ROM) alle Zeichen der zwei Zeichensätze fest zu verankern, von wo sie das Betriebssystem herausholen und auf den Bildschirm bringen kann.

Die Reihenfolge der Zeichen und ihr Code sehen Sie in der Tabelle 4. Sie ähnelt in mehreren Bereichen der ASCII-Reihenfolge, einige Spalten sind sogar identisch. Das macht eine Umrechnung – auch für das Betriebssystem – sehr einfach.

Folgende Blöcke der beiden Codearten entsprechen einander:

Ein Programm zur Umrechnung von ASCII-Code in Bildschirm-Code zeigt Listing 7.

Dabei habe ich als Variable gewählt:
- ACII = ASCII-Code
- Bild = Bildschirm-Code

In Bild 9 sind Bildschirm-Codes und entsprechende Funktionen nochmal aufgeführt:

Eine Zusammenfassung der Methoden, wie man in Basic Tasten abfragen kann.

Wir haben insgesamt vier Methoden kennengelernt und verwendet, um das Drücken einer Taste im Programm abzufragen:

1. Tastencode in Speicherzellen 203/653

```
10 A=PEEK(203)
20 B=PEEK(653)
30 IF A=ZEICHENCODE AND
      B=STEUERCODE THEN
      AKTION
```

2. ASCII-Code im Tastaturpuffer

```
100 POKE 198,0
110 A=PEEK(631)
120 IF A=ASCII-CODE THEN AKTION
```

3. Abfrage des ASCII-Codes mit GET/INPUT

```
200 GET A$
210 IF A$<>CHR$(ASCII-CODE) THEN AKTION 1
220 AKTION 2
```

4. Abfrage des Gänsefüßchen-Modus mit GET/INPUT

```
300 GET A$
310 IF A$ »ZEICHEN« THEN AKTION 1
320 AKTION 2
```

Diese vier Methoden haben alle eins gemeinsam:

Sie können immer nur eine einzelne Taste abfragen. Zwei oder gar mehrere Tasten gleichzeitig oder kurz hintereinander gedrückt ergeben keine sinnvollen Resultate.

Jetzt möchte ich Sie an meine allererste Darstellung (Bild 1 und 2) erinnern, nämlich wie die Tasten elektrisch angeordnet sind und wie das Betriebssystem sie abfragt. Ich möchte das hier noch einmal darstellen, erstens weil es eine gute Überleitung bildet zu meiner Methode der Vielfach-Tastenabfrage, zweitens weil diese Darstellung nicht vollständig war.

In Bild 10 ist noch einmal die VC 20-Tastenanordnung dargestellt, in Bild 11 diejenige des C 64. Trotz des Unterschiedes der elektrischen Anordnung ist bei beiden Computern die Abfragemethode identisch, nur die dafür benötigten Register haben unterschiedliche Adressen.

Zur Erinnerung:

Das Betriebssystem wählt der Reihe nach die senkrechten Spalten dadurch an, daß es die Zahl, die am Fuß jeder Spalte steht, in das Spalten-Register 37152 (beziehungsweise

BEDEUTUNG	ASCII-CODE	REVERSE DARSTELLUNG	FINKELN
LOCK (Sperre der Zeichensatz-Umschaltung	8	H	H
UNLOCK (Sperre aufheben)	9	I	I
RETURN	13	M	M
TEXT (2. Zeichensatz)	14	N	N
DEL (Zeichen löschen)	20	T	T
SHIFT RETURN (Cursor auf Anfang der nächsten Zeile)	141	[illegible]	SHIFT M
GRAF (1. Zeichensatz)	142	[illegible]	SHIFT N

Tabelle 3. Funktionen, die im Gänsefüßchen-Modus nur durch »Finkeln« eingetippt werden können

56320) schreibt. Diese Zahl ergibt in dualer Darstellung eine 0 an dieser Stelle.

Für jede Taste, die in der angewählten Spalte gedrückt ist, wird eine 0 in das andere Register 37153 (56321) geschrieben. Diese Dualzahl ergibt einen Dezimalwert, welcher aus dem Register herausPEEKbar ist.

Das, was das Betriebssystem macht, machen wir ihm nach, zuerst für den VC 20:

```
10 POKE 37152,247
20 PRINT PEEK (37152);PEEK(37153)
30 GOTO 10
```

und für den C 64:

```
10 POKE 56320,127
20 PRINT PEEK(56320);PEEK(56321)
30 GOTO 10
```

In Zeile 10 ist noch ein weiterer Unterschied zwischen den beiden Computern zu sehen. Beim VC 20 habe ich die Spaltenzahl 247 gewählt, weil in dieser Spalte die STOP-Taste liegt. Diese Spalte ist nämlich, wie wir ganz am Anfang ja schon festgestellt haben, die einzige Spalte, die wir mit Basic abfragen können. Bei POKEn der anderen sieben Spaltenzahlen in Zeile 10 wirft uns die 60mal in der Sekunde stattfindende Überprüfung der STOP-Taste aus dem Programm.

Beim C 64 ist das die Spalte 127, wie es uns ein Blick auf das Bild 11 zeigt.

Ich freue mich übrigens, daß ein aufmerksamer Leser aus Wien diesen Unterschied sofort bemerkt und mich darauf aufmerksam gemacht hat. Aber ich hatte damals nur einen VC 20 zur Verfügung, und mangels eigener Erprobung ist es mir nicht aufgefallen. Das hat sich übrigens jetzt geändert.

Diese Tastenabfrage hat den großen Vorteil, daß mehrere Tasten gleichzeitig drück- und abfragbar sind. Jede gedrückte Taste erzeugt eine 0 im »Reihen-Register«. Mit dem kleinen Programm oben können Sie es ausprobieren. Drücken Sie alle Tasten der Spalte 247 (127), die STOP-Taste bitte als letzte! Der rechte Zahlenstreifen auf dem Bildschirm zeigt die 0.

Lassen Sie die STOP-Taste (die 1-Taste) los, und es erscheint die 1, bei Loslassen der linken SHIFT-Taste (der ←-Taste) zusätzlich erhalten wir die 3. Das ist die Dezimalzahl für 00000011, die beiden 1er entstehen durch die losgelassenen Tasten.

Jede mögliche Tastenkombination in einer Spalte hat ihre spezielle und abfragbare (!) Codezahl im Register 37153 (56321), insgesamt 256 Kombinationen. Ist das nichts?

Diese Methode der Mehrfach-Tastenabfrage haben wir bereits erfolgreich eingesetzt.

Nun möchte ich diese Methode auf alle acht Spalten, also auf alle Tasten erweitern. Dazu müssen wir das oben erwähnte Hindernis, nämlich den Rausschmiß durch das Betriebssystem, überwinden. Dazu gibt es zwei Methoden. Methode 1 will ich nur kurz erwähnen – sie ist einen eigenen Aufsatz wert.

Man kann durch Beeinflussung der Speicherzellen 788 und 789 die Pause (Interrupt) zur Tastenabfrage des Betriebssystems künstlich verlängern und sie zur eigenen Abfrage benutzen.

Die zweite Methode ist einfacher (Listing 8). Wir schreiben das Programm oben (Zeilen 10 und 20) in Maschinensprache. Das läuft dann so schnell ab, daß es innerhalb zweier Unterbrechungen ausgeführt und somit vom Betriebssystem nicht gestört wird.

Da ich nicht annehme, daß alle Leser dieses Kurses in Maschinencode programmieren können, zeige ich es Ihnen einfach als Kochrezept. Es besteht aus einer Reihe von Zahlen, die über READ ... DATA in vorbestimmte Speicherplätze gePOKEt und von dort dann mit SYS gestartet werden.

Für den VC 20 gilt:

```
100 DATA 169, 254 ,141,32,145,173,33,145,141,255,
29,96
```

Für den C 64 gilt:

```
100 DATA 169, 254 ,141,0,220,173,1,220,141,255,
29,96
```

Ich habe einige Zahlen gekennzeichnet:

– Die zweite Zahl der DATA-Reihe in den Kästchen ist die Zahl der Spalte, die angewählt wird (ich habe 254 gewählt).

64er online Die 32 und 145 (0,220 beim C 64) ergeben die Registeradresse 37152 (56320), die DATA-Werte 33 und 145 (1,220) die Registeradresse 37153 (56321), die Werte 255,29 ergeben eine Speicherzelle 7679, auf die ich noch zurückkomme.

Die Registeradressen sind, wie immer bei Commodores 8-Bit-Computern, mit zwei Zahlen, das heißt mit 2 Byte dargestellt.

REGEL:
LSB + 256*MSB = Adresse
32 + 256* 145 = 37152
1 + 256* 220 = 56321
LSB = niederwertiges Byte
MSB = höherwertiges Byte

Diese 12 DATA-Zahlen, ich nenne sie X, werden eingelesen mit:

```
110 FOR K=1 TO 12
120 READ X
140 NEXT K
```

Damit diese Zahlen vom Basic-Programm nicht überschrieben oder gelöscht werden (was dasselbe ist), POKEn wir sie an das Ende des VC 20-Speichers ohne Erweiterung, nämlich ab Speicherzelle 7661. Für den C 64 geht es mit denselben Adressen natürlich allemal.

```
130 POKE 7660+K,X
```

Um ganz sicher zu gehen, daß mit dem Maschinenprogramm nichts passiert, schützen wir es, indem wir dem Computer vorgaukeln, daß sein für Basic zur Verfügung stehender Speicher bei Adresse 7659 aufhört. Diese Speichergrenze steht in den Speicherzellen 51/52 und 55/56, und sie kann natürlich durch POKEn anderer Zahlen willkürlich verändert werden.

Die entsprechenden Werte für die Grenze 7659 sind 235 und 29. Machen wir die Probe:

```
PRINT 235+256+29 (RETURN)
```
ergibt 7659.
```
20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29
```
Falls Sie diese Speicherzellen und ihre Anwendung nicht kennen, bitte ich um Zuschrift, da ich es hier aus Platzgründen nicht näher erläutern kann.

So, nun ist das Maschinenprogramm gespeichert und gesichert. Wir starten das Programm mit SYS-Aufruf der 1. Adresse und PEEKen danach das Register 37153 (56321), genau wie vorher.
```
150 SYS 7661
160 PRINT PEEK(37153): REM VC 20
160 PRINT PEEK(56321): REM C 64
170 GOTO 150
```
Nach RUN erzeugt der Rücksprung den schon obligatorischen Zahlenstreifen mit 255 für »keine Taste gedrückt«. Wenn wir die Taste 3 (RETURN beim C 64) drücken, dann muß entsprechend Bild 10 und 11 (Spalte 254) die Zahl 253 erscheinen. Das tut sie auch, aber immer wieder unterbrochen durch 255. Das kommt daher, daß das Auslesen des Registers 37153 (56321) in Basic doch schon zu langsam ist. Wir müssen daher die Funktion der Zeile 160 ebenfalls in das Maschinenprogramm einbauen, was ich in weiser Voraussicht in Zeile 100 schon vorbereitet habe. Der Trick ist nämlich dabei, daß der Inhalt des Registers vom Maschinenprogramm in eine vor dem bösen Basic geschützte Speicheradresse gebracht wird. Da wir ja einen gesicherten Bereich bereits haben, habe ich die Zelle 7679 ausgewählt.

Bitte ändern Sie die Zeile 160 ab:
```
160 PRINT PEEK(7679)
```
Jetzt haben wir's geschafft.

64ER ONLINE

Natürlich können wir mit diesem Programm alle acht Spalten abfragen. Wir brauchen bloß in Zeile 100 die Zahl, die hier im Kästchen steht, abändern, zum Beispiel in 253, und schon reagiert das Programm auf alle Tasten dieser Spalte.

Ich habe mir den Luxus erlaubt und unser Programm von oben erweitert, indem ich der Reihe nach alle acht Spalten aufrufe (siehe Listing 1). Dazu habe ich in Zeile 100 die Spaltennummer auf 0 gesetzt, was eigentlich unnötig aber einleuchtend ist, POKE aber dann jeweils die Spaltenzahl (in den Zeilen 200, 300, 400 etc.) auf diesen Platz, der die Adresse 7662 hat. Das Programm ist immer noch schnell genug, um innerhalb der zur Verfügung stehenden Zeit von 1/60 Sekunde alle Abfragen durchzuführen.

Ein Appell an alle 64er: Wenn Sie die Semikolons in den Zeilen 220, 320, 420 etc. (außer 920!!!) richtig plaziert haben, erscheinen die Zahlen für alle acht Spalten brav nebeneinander.

Beim VC 20 haben wir ein kleines Problem, hervorgerufen durch die kleinere Zeilenlänge, die uns eine Darstellung aller acht Zahlen nebeneinander nicht erlaubt.

Ich schlage deshalb vor, nach vier Zahlen ab Zeile 520 die Reihenfolge zu unterbrechen (Semikolon weglassen) und an den Anfang (Zeile 200) zurückzuspringen, es sei denn, die f-1-Taste wurde gedrückt (Zeile 560). Dann nämlich kommen die nächsten vier Zahlen dran – sogar in einer anderen Farbe. Eine weitere Tastenabfrage in Zeile 960, diesmal der .....-Taste – schauen Sie in der ASCII-Tabelle nach, welche Taste dieses Zeichen hat (Methode des waagrechten Sprunges zwei Spalten weiter) – entscheidet zwischen Wiederholung der zweiten Gruppe oder Rücksprung auf die ersten vier Zahlen. Wie gesagt, die Verwendung der Semikolons an der richtigen Stelle ist wichtig.

Jetzt ist die Gelegenheit da zum Experimentieren:
- mehrere Tasten aus verschiedenen Spalten
- mehrere Tasten aus derselben Spalte.

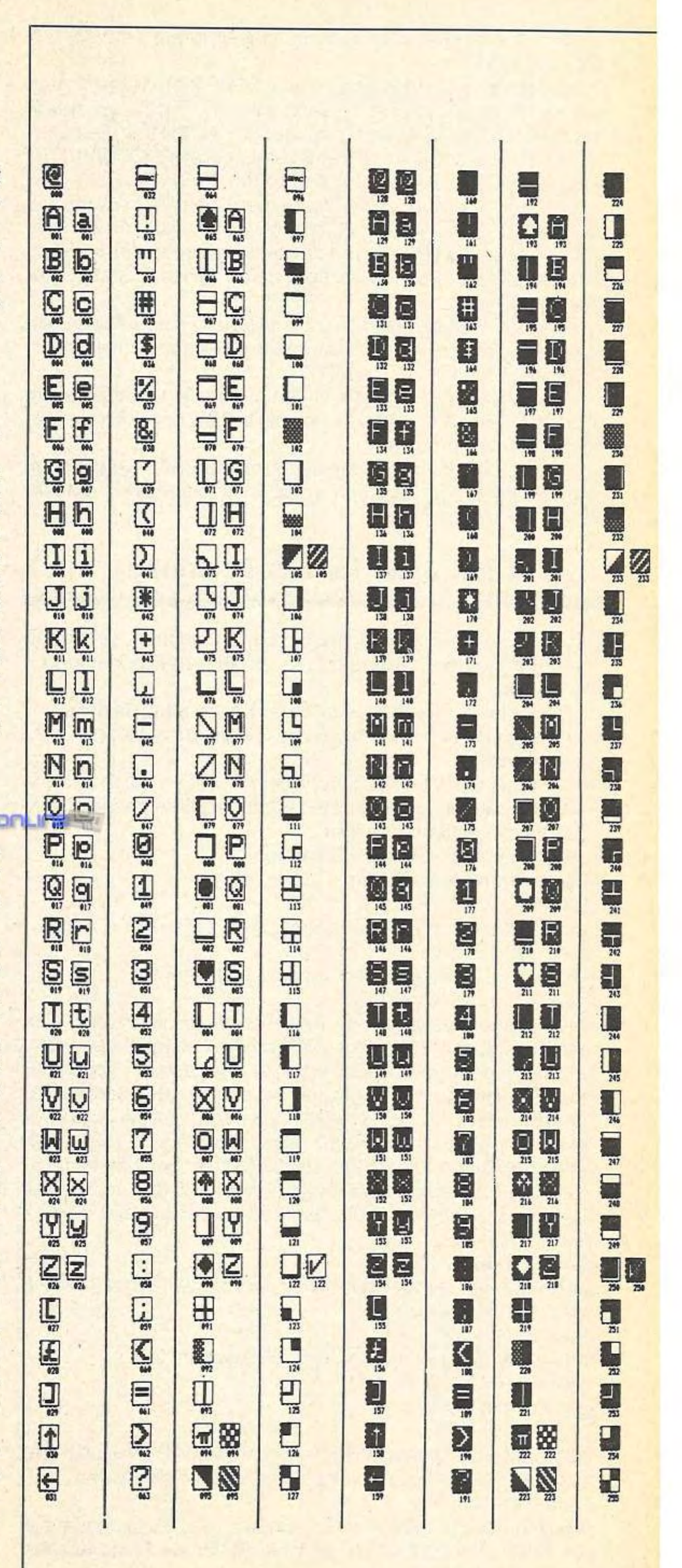


Tabelle 4. Bildschirm-Code

64er-online.de
64er-online.net

Ja, und irgendwann bleibt Ihnen das Programm mit BREAK IN .... stehen!!

Leider ist das schon wieder der STOP-Tasten-Effekt, den wir nicht ganz loswerden. Wenn nämlich Tasten der waagrechten Reihe 254 (beziehungsweise 127) gedrückt werden, interpretiert dies das Betriebssystem als STOP-Taste, die ja in dieser Reihe liegt, und stoppt das Programm.

Uns soll das aber nicht bedrücken, denn wir können in unserem Programm den Gebrauch dieser Tastenreihe einfach vermeiden. Für unsere Zwecke reichen alle anderen Tasten allemal aus. Man muß diese Einschränkung nur berücksichtigen.

Zusammenfassend sei gesagt, daß Sie in einem Programm jetzt alle Kombinationen aller Tasten abfragen können mit

```
IF PEEK(7679)= ..... THEN .....
```

Sie müssen sich lediglich die entstehenden Dualzahlen im Register 37153 (56321) in Dezimalzahlen umrechnen oder sie vorher ausprobieren.

Ich will auch am Schluß meiner Darstellung der Gewohnheit treu bleiben und das Kochrezept in einem kleinen Gericht anwenden (Listing 9).

Ein Spiel für vier Personen

Um mir und Ihnen die Sache leichter zu machen, verwende ich möglichst viele Teile aus den vorherigen Programmen.

Spielidee:

- Jeder Spieler wählt eine von vier bestimmten Tasten.
- Auf dem Bildschirm wird ein beliebiger Buchstabe »angesagt«
- Der Bildschirm füllt sich langsam mit Zufalls-Buchstaben
- Sobald der »angesagte« Buchstabe erscheint, sollen die Spieler ihre Taste drücken
- Wer zuerst drückt, hat gewonnen!

Programmbeschreibung:

Zeile 10 bis 50
schützt das Maschinenprogramm (wie vorher)

Zeile 100 bis 140
liest das Kochrezept »Maschinenprogramm« ein

Zeile 215 und Zeile 530
stellen einen besonderen Trick dar. Man nennt das eine »Flagge setzen« (set a flag). Die Abfrage der Tasten soll nämlich nur dann funktionieren, wenn ein Buchstabe (später VB genannt) mit dem anfangs »angesagten« Buchstaben BU übereinstimmt. Wenn VB=BU, dann wird in eine »sichere« Speicherzelle (ich habe 830 gewählt) eine Zahl (hier 250) hineingePOKEt. Diese Zahl in 830 wird bei der Abfrage ebenfalls abgefragt. Bei Beginn des Spieles muß diese Flagge natürlich eingeholt, das heißt auf 0 gesetzt werden (Zeile 215).

Zeile 220 bis 270
gibt Anweisungen und teilt den Spielern ihre Taste zu (von mir willkürlich ausgewählt)

Zeile 280/300
löscht den Bildschirm und färbt alles schwarz

Zeile 310/320
gibt Anweisungen

Zeile 330
wählt per Zufallsgenerator (wie vorher) einen Buchstaben BU aus; zwischen 1 und 10 (A bis J in Bildschirm-Code!)

Zeile 340/350
druckt diesen Buchstaben BU genau an das Ende des Textes der Zeile 320 (Bildschirmspeicher-Platz und Farbspeicher-Platz ausrechnen!)

Zeile 260/270 und 370/380
schalten den Text weiter mit irgendeiner Taste

Zeile 400
verzögert das Erscheinen des 1. Buchstabens

Zeile 520
wählt wieder per Zufallsgenerator einen Buchstaben, diesmal VB, aus (zwischen A und J)

Zeile 540
wählt per Zufallsgenerator einen Platz C auf dem Bildschirm aus, auf den der Buchstabe VB gePOKEt wird

Beim VC 20 fülle ich den ganzen Bildschirm von Platz 0 bis 505. Beim C 64 wäre das der Bereich von 0 bis 1000. Damit aber den Spielern wegen der viel kleineren Buchstaben des C 64 die Augen nicht übergehen, habe ich die eigentliche Formel C = INT(RND(0)*1000) in Zeile 540 so abgeändert, daß die Buchstaben nur auf jeden 2. Platz gePOKEt werden können.

Zeile 550/560
die Buchstaben VB werden in Weiß gePOKEt

Zeile 610
leiert die Tastenabfrage 15mal hintereinander durch. Diese Zahl bestimmt auch die Geschwindigkeit, mit der die Buchstaben auf dem Bildschirm erscheinen. Ihre Verkleinerung erhöht die Schwierigkeit des Spieles.

Zeile 620 bis 740
fragt die in Zeile 220 bis 250 definierten Tasten und die Flagge in 830 ab. Beim VC 20 ist jede der gewählten Tasten in einer anderen Spalte, daher sind vier POKEs in 7662 notwendig. Beim C 64 liegen je zwei Tasten in einer Spalte, daher nur zwei POKEs in 7662. Da aber gleichzeitige Tastendrücke vorkommen können (in einer Spalte natürlich), fragen Zeile 660 und 710 diesen speziellen Fall ab.

810 bis 840 beziehungsweise 850
meldet den Sieger (Aussprung aus der Abfrage)

Zusammenfassung

Diese »gleichzeitige« Mehrfach-Tastenabfrage ist genau genommen nicht ganz gleichzeitig.

Nichts in einem einzelnen Computer ist gleichzeitig! Alles erfolgt in einer Sequenz.

In dem Spielprogramm für vier Personen erfolgt die Tastenabfrage natürlich auch hintereinander, wodurch die zuerst abgefragten Tasten ihren Besitzern einen kleinen Vorteil gewähren, aber halt nur einen ganz winzigen!

Dieser Effekt wird durch das 15fache Durchlaufen der Abfrage und durch die Schnelligkeit des Maschinenprogrammes gemildert. Das ganze Programm in Maschinensprache geschrieben, würde natürlich durch die Geschwindigkeit auch die letzte Ungerechtigkeit ausmerzen.

So, liebe VC 20er und C 64er. Das war's.

Ich hoffe, Sie kennen jetzt die verschiedenen Codes und fühlen sich wohl bei der Tasten-Abfrage. Methoden und Kochrezepte haben Sie dazu ja jetzt genug.

Ich habe mich bemüht, Sie zum Experimentieren anzuregen, denn gerade diese spielhafte Auswirkung der Neugier unterscheidet den Amateur vom Profi und fördert das Verständnis der Zusammenhänge.

Ich habe mir am Anfang das Ziel gesetzt, allgemeinverständlich und ohne Fachchinesisch zu schreiben.

Ob das gelungen ist, können nur Sie mir sagen. Falls etwas unklar geblieben ist und Sie Fragen haben, schicken Sie sie mir, ich werde Ihnen antworten. Und wenn es sogenannte »gute Fragen« sind, schreibe ich vielleicht darüber den nächsten Kurs.

(Dr. Helmuth Hauck/cg)

Literatur:

1. VIC REVEALED von N. Hampshire, Computabits Ltd., 1981
2. M. Bassman, S. Lederman in COMPUTE!S FIRST BOOK OF VIC COMPUTE!, Books Publ., 1982
3. VC 20 SPIELE-BUCH 1 von A. Dripke, Computer Life Verlag, 1983
4. VC-INTERN von Angerhausen und Englisch, Data Becker, 1983
5. 64-INTERN von Angerhausen et al., Data Becker, 1983
6. DAS VC-20-Buch von M. Hegenbarth, M. Schäfer, Markt & Technik Verlag, 1983
7. VIC-20-PROGRAMMERS REFERENCE GUIDE von A. Finkel et al., Howard W. Sams & Co, 1982

Synthetische Steuerzeichen

Mit den »synthetischen Steuerzeichen« stehen Ihnen Möglichkeiten offen, die Sie bisher nicht für möglich gehalten haben: beispielsweise ein neuartiger Listschutz, Grafik und Breitschrift-Schmalschrift-Wechsel im Listing.

Nach dem Einschalten des VC 20 oder des C 64 und der »READY«-Meldung des Computers kann man sich nach Herzenslust auf dem Bildschirm austoben – zunächst ohne Programm. Sie können vorab Texte gestalten, Grafiken entwerfen, dabei die Farben wechseln, hier Zeichen ergänzen und da Stellen abändern. Komfortabel wird dieses Austesten der Möglichkeiten auf der »elektronischen Tafel« jedoch erst durch die Cursor-Steuertasten CLR/HOME, INST/DEL, RVS ON/OFF sowie durch die Farbwahltasten. Man erreicht damit ein hohes Maß an Flexibilität im Ansteuern jeder beliebigen Bildschirmstelle.

Schreibt man nun unversehens ein Anführungszeichen (engl.: quote) auf den Schirm, so verhält sich der Computer plötzlich anders: die Cursor-Tasten wirken nicht mehr – man kommt mit ihnen nicht mehr aus der Zeile heraus. Auch die Farbumschaltung mißlingt, das Bildschirmlöschen mit CLR/HOME versagt, lediglich DEL funktioniert noch. Und dies alles in voller Absicht! Warum? Statt der Ausführung der Steuerbefehle zieht es der Computer vor, sich die Anweisungen in Form reverser Steuerzeichen zu »merken«. Wozu? Nun, um Steuerbefehle dieser Art programmierbar zu machen. (Man stelle sich nur einmal vor, der Cursor sei nicht per Software steuerbar.) Nach dem ersten Anführungszeichen befindet sich der Computer offenbar in einem anderen Verarbeitungsmodus, genannt Quote-Modus, in dem er (mit Ausnahme von DEL) alle Steuerbefehle als Reverszeichen »speichert« und damit zur späteren Ausführung bereitstellt. Der Quote-Modus wird verlassen, sobald das zweite Anführungszeichen eingetippt oder die RETURN-Taste betätigt wird. Da die Steuerzeichen – eingeschachtelt in Gänsefüßchen – vom Computer wie Texte behandelt werden, können sie in Programmen auch als solche verarbeitet werden. Dazu stehen alle Stringoperationen zur Verfügung, die auch bei »normalen« Texten verwendet werden. Wirkung zeigen Steuerzeichen jedoch erst dann, wenn man sie mittels PRINT aktiviert. Fazit: Ein Basic-Listing kann normalerweise keine reversen Zeichen enthalten – es sei denn, es handelt sich um Steuerzeichen in Strings.

Steuerbefehle ohne »Gänsefüßchen«

Wir haben uns als Commodore-Anwender sicherlich schon längst an das reverse Q für CURSOR UP oder an das Herzchen für CLR/HOME gewöhnt. Doch es geht auch anders. Das muß es auch, wenn der Drucker zum Beispiel keine Steuerzeichen ausgeben kann. In diesem Fall bedient man sich der CHR$-Funktion, die durch die Befehlsfolge

```
PRINT CHR$(X)
```

aktiviert wird. Mit X = 147 wird beispielsweise ebenfalls der Bildschirm gelöscht. Wie kommt das? Tippen Sie bitte ein:
PRINT ASC("{CLR/HOME}")
Bitte tippen Sie die Ausdrücke, die in den geschweiften Klammern stehen, nicht als Buchstabenfolge ein, sondern zum Beispiel für PRINT ASC ("{CLR/HOME}") nach dem ersten Gänsefüßchen die Control-Taste und dann gleichzeitig die CLR-/HOME-Taste. Verfahren Sie bei den folgenden Beispielen dementsprechend.

Nach erfolgtem RETURN lesen Sie: 147. Das reverse Herzchen wird vom Computer also als Zeichen interpretiert, das den ASCII- beziehungsweise CHR$-Code 147 trägt. Ein weiteres Beispiel: Die Zeilen

```
PRINT " { CTRL-RVS ON} TEST"
```

und

```
PRINT CHR$(18)"TEST"
```

bewirken dasselbe, weil

```
PRINT ASC("{CTRL-RVS ON} ")
```

zeigt, daß der CHR$-Code des reversen R eben 18 ist.

Ausgehend von der Tatsache, daß jedem Steuerzeichen ein bestimmter Code in der CHR$-Liste (siehe Handbuch) zugeordnet ist, wurde die Idee geboren, daß diese Zuordnung – um es mathematisch auszudrücken – umkehrbar eindeutig sein müsse. Das heißt, zu jedem Steuerbefehl müßte auch ein entsprechendes reverses Steuerzeichen gehören, das denselben Zweck erfüllt.

Nun gibt es zum Beispiel die Möglichkeit, über

```
PRINT CHR$(14)
```

auf Kleinbuchstaben umzuschalten. Gibt es dafür auch ein Steuerzeichen? Auf konventionellem Weg hieße die Antwort klipp und klar: nein. Aber ...

Zu Beginn war alles nur Spielerei, bis sich die erstaunlichen Möglichkeiten und Anwendungen häuften. Deshalb: spielen Sie nun bitte mit bei den folgenden Tricks.

Steuerzeichen auf illegalem Weg

Bevor wir das große Geheimnis lüften, wollen wir erst einmal an ganz und gar legalen Steuerzeichen üben, wie man sie illegal eingibt. Halten Sie sich bitte zunächst streng an das angegebene Rezept, auch wenn es noch andere Eingabeformen gibt. Das gilt auch für die späteren Beispiele. Unsere erste Übung soll darin bestehen, den Bildschirm ohne CHR$(147) und ohne die Taste CLR/HOME zu löschen – im Direktmodus, versteht sich. Dazu geben Sie bitte hintereinander in einer Zeile ein:

1. Schritt:
   PRINT
2. Schritt:
   Erstes Anführungszeichen setzen. Jetzt befinden wir uns im Quote-Modus, den wir sofort wieder verlassen wollen. Daher:
3. Schritt:
   Zweites Anführungszeichen setzen.
4. Schritt:
   Mit DEL eine Stelle zurückgehen. Dabei wird das zweite Anführungszeichen gelöscht. Da wir uns nicht mehr im Quote-Modus befinden, reagiert der Computer auf Steuerbefehle.
5. Schritt:
   Mit CTRL-RVS ON auf Revers-Modus schalten. Das ist notwendig, da wir ja ein reverses Steuerzeichen quasi künstlich erzeugen wollen!
6. Schritt:
   SHIFT S eingeben. Beim geSHIFTeten S bekommen wir eines der S-Taste zugeordneten Grafikzeichen, in diesem Fall das gewünschte Herzchen. Nun verlassen wir den Revers-Modus:
7. Schritt:
   CTRL-RVS OFF. Wenn Sie wollen, können Sie noch ein Anführungszeichen zur optischen Abrundung anhängen.

Was jetzt auf dem Bildschirm steht, sieht so aus, als hätten

64er-online.de

wir nie den legalen Pfad verlassen, als hätten wir immer mit der CLR/HOME-Taste gearbeitet. Dieses Steuerzeichen ist jedoch künstlich entstanden – wird es auch wirken wie ein »echtes«? Das Drücken der RETURN-Taste überzeugt uns schnell davon, daß der Computer uns diese umständliche Manipulation nicht übelgenommen hat: es funktioniert tatsächlich!

Bedenken Sie bei der Eingabe, daß nur zwei Anführungszeichen auf dem Schirm erscheinen, obwohl insgesamt drei eingegeben wurden. Der Computer befindet sich also nach dem dritten Gänsefüßchen wieder im Quote-Modus.

An der Stelle, an der soeben mit SHIFT S das Steuerzeichen für CLR/HOME generiert wurde, kann nun jedes beliebige reverse Zeichen erzeugt werden. Diejenigen davon, die auch eine Steuerfunktion ausüben, aber nicht auf normalem Wege über direkte Tastendrücke erzeugt werden können, wollen wir »synthetische Steuerzeichen« nennen. Und nun folgen nützliche Beispiele mit diesen Steuerzeichen.

Die Synthetischen kommen

Wenn es – und das war die Frage, die alles ins Rollen brachte – anstelle der Umschaltung auf Kleinbuchstaben mit CHR$(14) ein äquivalentes Steuerzeichen gibt, dann muß es zwischem dem reversen E für WHT (Code 5) und dem reversen Q für CURSOR DOWN (Code 17) zu suchen sein. Numeriert man das Alphabet entsprechend durch, so findet man den Buchstaben N unter der Codezahl 14. Das reverse N müßte demnach das Steuerzeichen für die Umschaltung auf Kleinbuchstaben sein. Probieren wir es aus:

```
? ""{DEL RVS ON} N " {RVS OFF} →RETURN
```

64er online

Es klappt! Der Computer schluckt diesen ungewöhnlichen Befehl und führt ihn aus.

Dieser Erfolg motivierte eine Suche nach allen verfügbaren synthetischen Steuerzeichen; die Ausbeute fiel jedoch zunächst recht mager aus. Die nachfolgende Tabelle 1 zeigt Ihnen die synthetischen Zeichen und ihre Wirkungen bei Bildschirmausgaben (beim Drucker sieht's etwas anders aus – dazu aber später mehr).

haben und auf ihre Vorteile nicht mehr verzichten wollen – nicht nur, weil sie gegenüber der CHR$-Funktion Speicherplatz sparen.

So unscheinbar und unwichtig das SHIFT RETURN ist, so gewaltig sind die Möglichkeiten, die uns das zugehörige synthetische Steuerzeichen eröffnet. Es handelt sich um das reverse, geshiftete M (Code 141).

Das Super-Steuerzeichen

Gibt man »? " "{DEL}(TEST {CTRL-RVS ON} {SHIFT M} {CTRL-RVS OFF} LAUF"« ein, so zeigt sich nach dem Drücken der RETURN-Taste, daß der Computer nach dem Teilstring »TEST« einen Wagenrücklauf (carriage return) mit Zeilenvorschub (line feed) durchführt und »LAUF« direkt darunter ausgibt. Genau dieses Verhalten erwartet man von SHIFT RETURN. {SHIFT M} kann jedoch mehr – viel mehr! Das nächste Beispiel soll es beweisen:

```
10 ? ""{DEL} TEST {CRTL-RVS ON }{SHIFT  M} R {CRTL-RVS OFF} LAUF"
```

Geben Sie bitte dieses einzeilige Programm ein und starten Sie es. Wie erwartet erscheint »TEST« und darunter revers »LAUF«. Nun listen Sie bitte dieses Programm. Erstaunt? Das ist tatsächlich neu! Das Steuerzeichen für die Umschaltung in den Revers-Modus erscheint nicht im Listing, sondern wird entgegen aller bisherigen Kenntnisse ausgeführt. »LAUF« wird im Listing revers ausgegeben. Durch das SHIFT-RETURN-Zeichen wird der Basic-Interpreter offenbar veranlaßt, nachgestellte Steuerzeichen auch im Listing wirksam werden zu lassen. Es zeigt sich, daß diese Schlüsselfunktion alle Steuerbefehle, seien es Farbumschaltungen oder Cursor-Bewegungen, aktiviert. Ungeahnte Möglichkeiten eröffnen sich nun zur optischen Aufbereitung, das heißt Strukturierung und Gestaltung von Listings auf dem Bildschirm.

Beispiel 1:

Reverse REM-Zeile ohne sichtbares REM-Statement und ohne Zeilennummer

```
10 REM""{DEL} {CRTL-RVS ON} {SHIFT M} {SHIFT Q} R
{CRTL-RVS OFF} TESTLAUF
20 FOR I=1 TO 10
30 PRINT"TESTLAUF":NEXT
```

Steuerzeichen	CHR$-Code	Wirkung
H	8	Die manuelle Umschaltung zwischen Groß- und Kleinbuchstaben mit Hilfe der COMMODORE- und SHIFT-Tasten wird gesperrt.
I	9	Die obige Sperre wird wieder gelöst.
M	13	RETURN. Alles, was nach dem reversen M in der Zeile steht, wird nicht mehr gedruckt. Leider ist eine echte Simulation der RETURN-Taste zum Ändern von Basic-Zeilen unter Programmkontrolle nicht möglich.
N	14	Umschalten auf Kleinbuchstaben.
T	20	DEL. Hiermit können Sie Teile oder komplette Programmzeilen optisch verschwinden lassen. Dieses Steuerzeichen wirkt auch beim Listen eines Programms auf dem Schirm. Anwendung: partieller Listschutz.
SHIFT M	141	SHIFT RETURN. Das Super-Zeichen! Mehr dazu im Text.
SHIFT N	142	Umschalten auf Großbuchstaben.
SHIFT T	148	INS. Kann zum Ändern von Text an bereits gedruckten Text benutzt werden. Beispiel: ? "{SHIFT T}X" liefert ein X auf dem Schirm. Bei mehrfacher Abarbeitung wird stets ein weiteres X an die bereits vorhandenen angehängt.

Tabelle 1. Steuerzeichen und ihre Wirkung

Die Zeichen dieser Tabelle müssen als reverse Zeichen wie oben erzeugt werden (Ausnahmen: T für DEL und SHIFT T für INS).

Auch wenn die Eingabe der synthetischen Steuerzeichen umständlich erscheint, man wird sich schnell daran gewöhnt

Die Zeilen 20 und 30 sind Beiwerk, damit die REM-Zeile nicht so allein dasteht. Listen Sie das Programm bitte. Sie sehen die reverse Überschrift »TESTLAUF« ohne REM und ohne Zeilennummer und darunter wie gewohnt die Programmzeilen 20 und 30. Wie funktioniert dieser Trick? Das reverse

{SHIFT M} bewirkt einen Sprung zum Anfang der nächsten Zeile und gibt die nachfolgenden Steuerzeichen zur Ausführung frei. Mit dem reversen {SHIFT Q} geht der Cursor eine Zeile nach oben, also auf »1« der Zeilennummer 10. Reverses R schaltet auf reverse Zeichen um, in denen dann der REM-Text »TESTLAUF« ausgegeben wird. Dabei überschreibt der Computer die Zeilennummer sowie das REM-Statement. Das alles vollzieht sich so schnell, daß es sich der Beobachtung entzieht. Der Anwender sieht nur noch die reverse Überschrift und kann demzufolge an dieser Zeile auch nichts mehr ändern.

Sollten Sie die Original-REM-Zeile noch auf dem Bildschirm haben, so hängen Sie an das »F« von »TESTLAUF« noch das Steuerzeichen für CURSOR DOWN an. Sie werden sehen, daß beim Listen sogar eine Leerzeile zwischen Überschrift und Programm entsteht. Selbstverständlich können derartige REM-Zeilen beliebig im Programm verstreut sein. Wirkung: Im Programmlauf werden unter der reversen Überschrift »Quadratzahlen« die Ergebnisse ausgegeben, während beim Listen des Programms der PRINT-Befehl verschwiegen wird und man eine REM-Zeile wie in Beispiel 1 vermutet. Damit trägt eine PRINT-Zeile zur optischen Strukturierung eines Listings bei - ein Effekt, der bislang in dieser Form unmöglich schien. Nun einige Erläuterungen zu den Beispielsprogrammen 2 bis 4:

Beispiel 2. Farbige REM-Zeile (ändern Sie nur Zeile 10).

Beispiel 3. Wirkung: Grüne Überschrift ohne Zeilennummer und ohne REM, gelbes Listing.

Beispiel 4. Wirkung: Die bekannten Herzchen sorgen nach dem Listen jeder Zeile für ein Löschen des Bildschirms, so daß die beiden Programmzeilen weder einzeln noch insgesamt sichtbar bleiben und die Codezahlen A und B dem Anwender vorenthalten werden.

Beispiel 2:
Farbige REM-Zeile (ändern Sie nur Zeile 10)

```
10 REM ""{DEL} {CRTL-RVS ON} {SHIFT M} {SHIFT Q} {↑}
{CRTL-RVS OFF} TESTLAUF {CRTL-RVS ON {SHIFT π}
{CRTL-RVS OFF}
```

Beispiel 3:
Listschutz

```
10 A=4711 : REM""{DEL}{CRTL-RVS ON}{SHIFT M}
{SHIFT S}
20 B=0815 : REM""{DEL}{CRTL-RVS ON}{SHIFT M}
{SHIFT S}
```

reverses SHIFT M für SHIFT RETURN
reverses SHIFT S für CLR/HOME

Beispiel 4:
REM-Ersatz

```
10 PRINT""{DEL}{CRTL-RVS ON}{SHIFT M}{SHIFT S}R
{CRTL-RVS OFF}QUADRATZAHLEN
```

reverses SHIFT M für SHIFT RETURN
reverses SHIFT S für CLR/HOME
reverses R für RVS ON

```
20 FOR I=1 TO 10
30 PRINT I,I*I : NEXT
```

Wirkung: Im Programmlauf werden unter der reversen Überschrift »Quadratzahlen« die Ergebnisse ausgegeben, während beim Listen des Programms der PRINT-Befehl verschwiegen wird und man eine REM-Zeile wie in Beispiel 1 vermutet. Damit trägt eine PRINT-Zeile zur optischen Strukturierung eines Listings bei - ein Effekt, der bislang in dieser Form unmöglich schien.

Sie werden sicherlich Spaß an der Vielseitigkeit dieser neuen Methode gewinnen und eine Vielzahl von Anwendungen austüfteln.

Bisher wurde die Erzeugung synthetischer Steuerzeichen und ihre Wirkung bei Bildschirmbetrieb dargestellt. Jetzt soll von neuen Möglichkeiten für Druckeranwender die Rede sein. Dabei wird auf den Drucker VC 1515 Bezug genommen, mit dem die Steuerzeichen ausgetestet wurden. Es ist nicht auszuschließen, daß der eine oder andere Druckertyp unterschiedlich reagieren wird. Die wichtigsten Steuerfunktionen dürften jedoch auf allen Geräten die gleichen Reaktionen hervorrufen.

Benutzen Sie die synthetischen Steuerzeichen, um Ihre Druckerlistings übersichtlicher zu machen.

Bevor es nun zur Sache geht, ein kurzer Abriß des vorangegangenen Abschnitts:

Zur Cursor-Steuerung für RVS ON/OFF oder zum Beispiel für Farbumschaltungen stehen dem VC 20- beziehungsweise C 64-Anwender zwei Möglichkeiten zur Verfügung. Erstens kann die CHR$-Funktion genutzt werden (Beispiel ?CHR$(19) bewirkt HOME) und zweitens erlauben reverse Steuerzeichen eine direkte und kurze Eingabe der Steuerbefehle (Beispiel: das reverse S veranlaßt ebenfalls HOME). Nun existieren jedoch einige CHR$-Codes, für die äquivalente Steuerzeichen nicht über zugehörige Tasten abgerufen werden können. So schaltet CHR$(14) beispielsweise auf Kleinbuchstaben um. Mit Hilfe der ASC-Funktion zeigt sich, daß das reverse N den CHR$-Code 14 trägt. Damit kann dieses Zeichen ebenfalls zur Steuerung herangezogen werden. Da außer den konventionellen Steuerzeichen keinerlei reverse Symbole in Strings vorkommen können, muß das reverse N quasi künstlich erzeugt werden (daher der Name »Synthetische Steuerzeichen«). Aber das ist kein Problem! Wir haben bereits ein Eingabeverfahren kennengelernt, das ich aber jetzt nicht wiederholen möchte. Statt dessen möchte ich Ihnen ein anderes Verfahren zeigen, das vielleicht ein wenig übersichtlicher ist als das bereits beschriebene.

Geben Sie die Programmzeile, die ein im String stehendes Steuerzeichen erhalten soll, wie gewohnt ein. Reservieren Sie dabei jedoch mittels Space die Stelle im Textstring, wo später das synthetische Steuerzeichen stehen wird. Schließen Sie die Eingabe der Zeile mit Betätigung der Return-Taste ab. Nun können sie ohne Schwierigkeiten den Cursor auf die bewußte Stelle bewegen, durch gleichzeitiges Drücken von CTRL und RVS ON in den Revers-Mode schalten (es erscheint kein reverses R!) und schließlich den freigehaltenen Platz mit dem entsprechenden Reversezeichen belegen (in unserem Beispiel also mit dem reversen N). Die so ergänzte Programmzeile wird jetzt durch erneutes Drücken der Return-Taste verlassen. Fertig. Auf diese Weise lassen sich sowohl die altbekannten als auch die neuen Steuerzeichen leicht erzeugen und entsprechend ins Programm einfügen. In den weiteren Ausführungen werde ich die synthetischen Steuerzeichen vereinfachend in geschweiften Klammern darstellen (zum Beispiel reverses N = {N}).

Die Drucker-»Synthies«

Nun zum eigentlichen Thema: Man könnte fast wetten, daß einige Druckerbesitzer unter unseren Lesern inzwischen beim Experimentieren mit synthetischen Steuerzeichen nicht schlecht gestaunt haben. Denn auch hier eröffnen sich neue Möglichkeiten, an die bislang nicht zu denken war. So sind zwar keine spektakulären Effekte in laufenden Programmen zu erzielen, dafür jedoch hat man erstmals die Möglichkeit, Druckerlistings zu manipulieren. Zuvor möchte ich Ihnen jedoch eine Liste der Steuerzeichen (Tabelle 2) geben. Es ist zu beachten, daß die mit *) gekennzeichneten Steuerzeichen nicht zu den synthetischen zählen, da sich diese direkt über Tasten eingeben lassen. Sie wurden nur der Vollständigkeit halber mit in die Tabelle aufgenommen.

64er-online.de

Steuerzeichen	CHR$-Code	Wirkung
{H}	8	Umschaltung auf Grafik-Modus
{J}	10	Zeilenvorschub
{M}	13	RETURN. Nachfolgende Zeichen werden nicht mehr gedruckt.
{N}	14	Umschaltung auf Breitschrift
{O}	15	Umschaltung auf normale Schriftbreite
{P}	16	Festlegung der Druckstartposition
*){Q}	17	Umschaltung auf KLeinbuchstaben
*){R}	18	RVS ON
{Z}	26	Zeichenwiederholung (Grafik)
{[}	27	Punktadresse für Druckstart
{SHIFT M}	141	SHIFT RETURN. Nachfolgende Zeichen werden in einer neuen Zeile gedruckt.
*){SHIFT Q}	145	Umschaltung auf Großbuchstaben
*){SHIFT R}	146	RVS OFF

Tabelle 2. Liste der Steuerzeichen

Nun zu den Erläuterungen derjenigen Zeichen, die sinnvolle Anwendungen erlauben.

Grafik im Listing

Das reverse H gestattet die Umschaltung des Druckers in den Grafik-Modus. Alle nachfolgenden Zeichen im String werden – sofern ihr jeweiliger CHR$-Code größer als 127 ist – als Grafikinformationen interpretiert. In diesem Modus bleibt der Drucker so lange, bis er mit {N} oder {O} auf Schriftbetrieb zurückgeschaltet wird. Betrachten Sie bitte Beispiel 1.

Zwischen »TEST« und »LAUF« befindet sich in Bild 2 ein selbstdefiniertes Grafikzeichen, das sowohl im Programmlauf als auch im Listing ausgegeben wird. Auch wenn Sie es zunächst nicht glauben – Sie sehen ein Original-Listing und nicht etwa gePRINTete Textzeilen. Ich will Ihnen den Trick verraten. In einer Vergrößerung sieht das Grafiksymbol aus wie im Bild 3 gezeigt.

Die Umrechnung des dualen Bitmusters in Dezimalzahlen liefert diejenigen CHR$-Codes, durch die das Zeichen spaltenweise beim Drucken entsteht. Man muß also nur wiederum synthetische Zeichen finden, die die gesuchten CHR$-Codes tragen. (Tabelle 3).

"TEST {H}	{SHIFT H}	{SHIFT MINUS}	{SHIFT T}	{SHIFT T}	{SHIFT T}	
8	136	156	148	148	148	
{SHIFT T}	{SHIFT T}	{SHIFT MINUS}	{SHIFT H}	{SHIFT *}	{SHIFT *}	{O}LAUF "
148	148	156	136	128	128	15

Tabelle 3. Aus dem Beispiel in Bild 3 ergeben sich diese Charakterstring-Codes.

Mit dieser Folge von reversen Zeichen erzielen Sie also den Ausdruck, der im Bild 2 abgebildet ist.

Folgt dem {H} keine Grafikinformation, das heißt ist der CHR$-Code kleiner als 128, so kommt es zum Abbruch des Listings. Diese Tatsache kann man sich bewußt zunutze machen, wenn man ohne größeren Programmieraufwand einen Drucker-Listschutz in sein Programm einbauen will. Der Bildschirmbetrieb leidet darunter nicht, da {H} dort nur eine Verriegelung der Umschaltung zwischen Groß- und Kleinschrift bewirkt.

Zeilenvorschub

Die Beispiele in Bild 4 und 5 zeigen, daß {J} einen Zeilenvorschub mit Rückwagenlauf verursacht. In Bild 4 befindet sich das reverse J zwischen »TEST« und »LAUF«, bei Bild 5 am Schluß des Strings. Da man das zweite Anführungszeichen eines Strings weglassen kann, sofern keine weiteren Basic-Befehle mehr in dieser Zeile folgen sollen, ergibt sich so die Möglichkeit, echte Leerzeilen mit {J} im Listing (Bild 4) zu erzeugen.

Breitschrift - Schmalschrift

Während {N} beim Bildschirmbetrieb auf Kleinbuchstaben umschaltet, zeigt der Drucker eine andere Wirkung. Er wird veranlaßt, alle nachfolgenden Zeichen in doppelt breiter Schrift auszugeben. Es ist also bei der Anwendung von {N} Vorsicht geboten, da ein für den Bildschirm konzipiertes Programm beim Druckerlisting unerwünschte Steuerungen zur Folge haben kann. In diesen Fällen sollte man {N} wieder durch den konventionellen Befehl CHR$ ersetzen.

Wollen Sie jedoch die Breitschrift zur übersichtlichen Gestaltung eines Listings nutzen, so können Sie ruhig das {N} hier und da im Programm verstecken. Es bleibt demjenigen, der das Programm lesen soll, sowieso verborgen. Wenn nur eine Hervorhebung einzelner Worte, Befehle oder Zeilen gewünscht wird, muß die Breitschriftphase entsprechend mit {O} beendet werden. Im Beispielprogramm in Bild 6 sehen Sie dieses Vorhaben realisiert. Hier bewirkt das {N} vor dem Wort »Test« die Umschaltung auf Breitschrift, während ein {O} vor »Lauf« diesen Modus beendet.

Die Druckerstartpositionierung – vergleichbar mit TAB – wird normalerweise mit CHR$(16) und nachgestellter Startadresse vorgenommen. Also etwa:

```
PRINT#4,CHR$(16) "10 TEST"
```

Diese Zeile druckt beginnend in Spalte 10 nur das Wort »Test« aus. Die im String vorangestellte Zahl dient dabei als Startadresse. Das kürzere Äquivalent sieht nun wie folgt aus:

```
PRINT#4,"{P} 10 TEST"
```

Doch Vorsicht! Obwohl diese Anweisung während des Programmlaufs korrekt ausgeführt wird, verschluckt sie der Drucker im Listing vollständig. Das Beispiel in Bild 7 zeigt, daß von {P}10 zwischen »TEST« und »LAUF« nicht mehr zu bemerken ist.

Damit verbietet sich die Anwendung des synthetischen {P} bei Programmen, die zum Beispiel zur Veröffentlichung vom Drucker dokumentiert werden sollen.

Für die Zeichen {Z} Zeichenwiederholung und {[} (Punktadresse für Druckerstart) haben sich bislang noch keine sinnvollen Einsatzmöglichkeiten ergeben. Sie sollen daher jetzt nicht weiter diskutiert werden. Auch auf die Beschreibung von {Q}, {R}, {SHIFT Q} und {SHIFT R} möchte ich verzichten, da sie keine synthetischen Zeichen sind und sich damit »ganz normal« verhalten.

Ich persönlich benutze die synthetischen Steuerzeichen gern zur Gestaltung von Listings. Wie so etwas aussehen kann, habe ich Ihnen im abschließenden Beispiel (Bild 8) zusammengestellt.

Vielleicht versuchen Sie einmal, die versteckten und unsichtbaren Steuerzeichen herauszufinden.

Die Suche nach den Synthetischen

Das folgende Programm ermöglicht die systematische Suche nach allen vom Betriebssystem unterstützten Steuerzeichen (Listing 1). Man ist nun nicht mehr angewiesen auf zum Teil lückenhafte Tabellen synthetischer Steuerzeichen, sondern kann sich statt dessen selbst auf die Suche begeben.

Das Programm-Listing zur Steuerzeichensuche (Listing 1) gibt nach Eingabe des gewünschten ASCII-Wertes alle Tastenkombinationen aus, deren Betätigung die Tastaturdecodierroutine dazu veranlaßt, den entsprechenden ASCII-Wert in den Tastaturpuffer zu schreiben.

```
0   0   0   0   0   0   0   0   0   0   0     1
0   0   0   0   0   0   0   0   0   0   0     2
0   ●   ●   ●   ●   ●   ●   ●   0   0   0     4
●   ●   0   0   0   0   0   ●   ●   0   0     8
0   ●   ●   ●   ●   ●   ●   ●   0   0   0    16
0   0   0   0   0   0   0   0   0   0   0    32
0   0   0   0   0   0   0   0   0   0   0    64
                                           +128
------------------------------------------------
136 156 148 148 148 148 148 156 136 128 128
```

Bild 3. Vergrößertes Grafiksymbol aus dem Beispiel in Bild 1

```
TEST⇔LAUF

10 REM GRAPHIK-MODUS
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST⇔LAUF"
50 PRINT#4:CLOSE4

READY.
```

Bild 2. Beispiel zum Grafik-Modus

```
TEST LAUF
10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST
LAUF"
50 PRINT#4:CLOSE4
READY.
```

Bild 4. Beispiel-Programm zum Zeilenvorschub im Wort

```
TESTLAUF
10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF

50 PRINT#4:CLOSE4
READY.
```

Bild 5. Beispiel-Programm: Zeilenvorschub am Wortende

```
TESTLAUF

10 REM BREITSCHRIFT
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Bild 6. Beispiel zum Umschalten von Schmal- auf Breitschrift

```
TEST      LAUF

10 REM DRUCKSTARTPOSITION
15 REM IST IM LISTING
20 REM NICHT SICHTBAR
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Bild 7. Beispiel: Ab Spalte 10 wird gedruckt. Die Druckeranweisung wird nicht gelistet.

```
100 REM DEMO ZUR OPTISCHEN STRUKTURIERUNG
110 REM      VON DRUCKER-LISTINGS MIT
120 REM"     SYNTHETISCHEN
130 REM"     STEUERZEICHEN
200 REM"*  HAUPTPROGRAMM  *
210 FORI=1TO64
220 PRINT"64'ER-MAGAZIN
230 GOSUB300:NEXT:"
240 END
300 REM"*  UNTERPROGRAMM  *
310 FORJ=1TO50:NEXT
320 RETURN:"
400 REM"ORIGINAL-LISTING!"
READY.
```

Bild 8. Beispiel zur Gestaltung von Listings durch synthetische Steuerzeichen

Es werden allerdings auch solche Kombinationen aufgeführt, deren »ASCII-Werte« zwar in den Decodiertabellen (die ab $EB81 im Betriebssystem beginnen) an entsprechender Stelle aufgeführt sind, jedoch NICHT im Tastaturpuffer erscheinen (zum Beispiel die »2« bei Betätigung der SHIFT-Taste). Der Leser wird diese jedoch schnell von der ersten Gruppe unterscheiden können.

Zum Schluß noch zwei Bemerkungen:

1) Bei gleichzeitigem Erscheinen von zwei oder mehreren Tastenkombinationen für einen bestimmten ASCII-Code kann oft – jedoch nicht immer – eine Kombination gegen eine andere ausgetauscht werden, ohne das Endresultat zu verändern (Beispiel für eine Ausnahme: Sowohl die RUN/STOP-Taste wie auch die CTRL C-Kombination belegen gemäß der Decodiertabellen den ASCII-Wert »3«, jedoch kann mit »CRTL-C« kein Programm gestoppt werden, da für die Abfrage der RUN/STOP-Taste eine gesonderte Routine zuständig ist, die nur diese Taste erkennt).

2) Alle »synthetischen Steuerzeichen«, für deren Erzeugung der Basic-Interpreter zuständig ist, kann dieses Programm nicht erkennen, da es lediglich auf die Tastaturdecotiertabellen im Kernel Bezug nimmt.

Da das Programm lediglich die Tastaturdecodiertabellen des Betriebssystems benötigt (und die dort aufgeführten ASCII-Werte in ein ARRAY einliest) kann es leicht durch entsprechende Abänderung der in Zeile 20 enthaltenen Anfangsadressen in eine auch auf dem VC 20 laufende Version umgeschrieben werden.

(Jürgen Wagner/Engin Gülen/cg)

64'er online

```
1 REM          *  STEUERZEICHEN   *
2 REM           *  FUER DEN CBM-64 *
3 REM            *  ENGIN  GUELEN   *
4 REM             *  POSTWEG     2   *
5 REM              *  4192 KALKAR 1    *
9 POKE53280,0:POKE53281,0
10 DIMAS(3,63)
18 REM ANFANGSADRESSEEN DER
19 REM TASTATURDEKODIERTABELLEN
20 DATA60289,60354,60419,60536
30 FORI=0TO3:READA(I):NEXT
40 FORI=0TO3:FORJ=0TO63
50 AS(I,J)=PEEK(A(I)+J)
60 NEXTJ,I
159 REM TASTATURMATRIX
160 DATADEL,RETURN,CRSR(RIGHT),F7,F1,F3,F5,CRSR(DOWN)
161 DATA3,W,A,4,Z,S,E,SHIFT(L)
162 DATA5,R,D,6,C,F,T,X
163 DATA7,Y,G,8,B,H,U,V
164 DATA9,I,J,0,M,K,O,N
165 DATA+,P,L,-,.,DOPPELPUNKT,@,KOMMA
166 DATA£,*,;,HOME,SHIFT(R),=,↑,/
167 DATA1,←,CTRL,2,SPACE,C=,Q,RUN/STOP
170 DIMTA$(64):FORI=0TO63:READTA$(I):NEXTI
180 DATA-----,SHIFT,C=    ,CTRL  ,
190 FORI=0TO3:READA$(I):NEXT
200 INPUT"■■■■■■■■■■■ ASCII-KODE ";AS%
210 IFAS%<0ORAS%>255THEN200
220 PRINT:PRINT:FORI=0TO3:FORJ=0TO63
230 IFAS%=AS(I,J)THENPRINT"■"A$(I),:GOTO250
240 GOTO270
250 IFLEFT$(TA$(J),2)=LEFT$(A$(I),2)THENPRINT:GOTO270
260 PRINT"◣"TA$(J)
270 NEXTJ,I
280 PRINTSPC(193)"██ TASTE "
290 GETA$:IFA$=""THEN290
300 GOTO200

READY.
```

Listing 1. Steuerzeichensuche

64er-online.de
64er-online.net

So macht man Programme schneller

Basic-Programme können nach drei Gesichtspunkten optimiert werden: strukturiert und lesbar, schnell sowie speicherplatzsparend. In diesem Beitrag wird der Aspekt der Laufgeschwindigkeit und ihrer Verbesserung behandelt.

Es gibt einige Dinge auf der Welt, die man sehr wohl einzeln, aber nicht alle gleichzeitig haben kann. Wirtschaftswachstum, keine Inflation und niedrige Arbeitslosigkeit lassen sich eben nicht unter einen Hut bringen.

Bei der Computerei, oder genauer gesagt beim Programmieren, gibt es in einer höheren Sprache wie Basic ebenfalls so ein magisches Dreieck: strukturierte Programme, minimaler Speicherbedarf, kürzere Laufzeiten.

Gut lesbare und klar gegliederte Programme brauchen oft viel Speicherplatz. Deswegen soll dieser Beitrag Sie zum Experimentieren anregen, mit welchen Methoden Basic-Programme schneller gemacht werden können.

Basic ist nicht immer gleich Basic

64er online

Die Commodore-Handbücher sagen leider zu diesem Thema recht wenig. Ich bitte Sie, an Ihrem C 64 Platz zu nehmen.

Als erstes wollen wir uns einen einfachen aber typischen Programmablauf überlegen, welchen wir mit mehreren Basic-Möglichkeiten programmieren können. Übrigens: Wegen der guten Lesbarkeit schreiben Sie die nachfolgenden Basic-Zeilen mit Leerstellen zwischen den Befehlen.

Ich weiß, es geht auch kürzer, aber bei meinen Experimenten spielt Speicherplatz keine Rolle und außerdem habe ich noch einen Hintergedanken, den ich erst später erklären kann. Nicht zuletzt will ich dadurch auch erreichen, daß die Laufzeiten der Programme mit den Ihren übereinstimmen.

Schalten Sie bitte auch Programmierhilfen (Toolkit, Simon's Basic etc.) und Disk Operating-Programme (DOS 5.1) aus, denn sie verlangsamen den Programmablauf.

Die interne Uhr mißt die Zeit

Der C 64 hat eine interne Uhr, deren Zeit abgefragt, ausgedruckt und somit zur Zeitmessung verwendet werden kann. Im Befehlssatz der Commodore-Handbücher finden Sie dazu die beiden Funktionen TI und TI$. Mein Zeitmessungsprogramm besteht aus zwei Zeilen.

Die »Stoppuhr« wird gestartet mit: `10 TI$="000000"`

Sie wird am Ende des Testprogramms gestoppt mit:

```
1000 PRINT TI/60 " SEKUNDEN" :END
```

Zwischen diese beiden Zeilen stecken wir dann die Prüflinge, das heißt die Programme, deren Laufzeit wir messen wollen. Zur Prüfung geben wir zuerst die Zeile 15 mit einer simplen Verzögerungsschleife ein.

```
15 FOR T=1 TO 100:NEXT T
```

Dieses Programm hat eine Laufzeit von 0,13 Sekunden.

Das erste Testprogramm

Löschen Sie bitte wieder die Zeile 15. Als Programm, welches wir in mehreren Versionen programmieren wollen, habe ich mir einen Ablauf ausgesucht, der auch optisch verfolgbar ist. Auf dem Bildschirm soll nämlich der Buchstabe A gleich 374mal nebeneinander gedruckt werden.

Die Zahl 374 hat nichts Magisches an sich. Es sind ganz einfach 17 Zeilen voller A's auf dem Schirm. 17 Zeilen lassen uns genug Platz, um die gestoppte Zeit darunter gut lesbar anzuzeigen.

In der **Version 1** des Programms verwenden wir POKE-Befehle, mit denen wir das A und die dazugehörige Farbe auf den Bildschirm, das heißt in den Bildschirmspeicher (Video-RAM) und den Farbspeicher (Color-RAM) bringen. Beim C 64 sind es die Speicherzellen 1024 und 55296. Schauen Sie bitte in den Commodore-Handbüchern nach und vergewissern Sie sich, daß Sie dieses System des Zeichen-POKEns verstehen. Es ist dort gut beschrieben. Der Buchstabe A hat den Bildschirm-Codewert 1. Als Farbe wähle ich die »Normalfarben« der beiden Computer. Der Farbcode für das Blau ist 14. Das Programm beginnt mit dem Löschen des Bildschirms (Zeile 20) und POKEt dann in Zeile 40 das erste A in den ersten Platz des Bildschirms.

```
20 PRINT CHR$ (147)
40 POKE 1024+Z:POKE 55296+Z,14
```

In Zeile 40 finden Sie zusätzlich eine Variable »Z«. Wie geplant, soll das A 374 mal gePOKEt werden. Also müssen wir die Zahl der Speicherzelle laufend um 1 erhöhen. Dazu erfinden wir diese Variable Z, die zur Speicherzelle addiert wird. Wir setzen Z am Anfang des Programms (in Zeile 30) auf Null und zählen es in Zeile 50 um 1 weiter.

```
30 Z=0
50 Z=Z+1
```

Als nächstes müssen wir prüfen, ob Z den Schlußwert 374 erreicht hat. Wenn nicht, dann soll der nächste POKE-Befehl ausgeführt werden, das heißt wir springen auf Zeile 40 zurück. Dann kommt die Zeile 1000 zum Zug mit dem Stoppen und Ausdrucken der Laufzeit. Also:

```
60 IF Z=374 THEN 1000
70 GOTO 40
```

Ich schlage vor, daß Sie das kleine Programm noch mal LISTen, damit wir es komplett sehen können.

Die Laufzeit wird nur davon beeinflußt, was zwischen den Zeilen 10 und 1000 steht. Sie können vor der Zeile 10 dem Programm hinzufügen, was Sie wollen.

Ein erster Probelauf mit RUN bringt das gewünschte Ergebnis, nur eins ist noch unschön: Der PRINT-Befehl in Zeile 1000 druckt uns die Zeit oben in die 2. Zeile, wo sie schlecht erkennbar ist. Wir könnten sie in einer anderen Farbe drucken, aber ich habe einen besseren Vorschlag.

Ersatz für den Befehl »PRINT AT«

Wir brauchen ein Kochrezept, um mit einem PRINT-Befehl an einen ganz bestimmten Platz auf dem Bildschirm drucken zu können. Einige Basic-Dialekte kennen den Befehl »PRINT-AT«. Welche Möglichkeiten bietet uns das Basic von Commodore?

1) PRINT"{Cursor Down}{Cursor Right}"
2) PRINT TAB(X)
3) PRINT SPC(X)

Um zum Beispiel an den 3. Platz in der 20. Zeile die Zeit zu drucken, müßten wir 18mal das inverse Q für Cursor Down und 1mal Cursor Right eingeben.

Mit TAB (X) geht es besser. Wir haben nur ein Problem, daß nämlich der höchste zulässige Wert für X nur 255 ist (X nennt

man das »Argument«). Wir müssen deshalb zwei TAB-Befehle hintereinander setzen, um einen Abstand von 400 Leerstellen zu erzeugen.

```
1000 PRINT TAB (255) TAB (155) TI/60"SEKUNDEN":END
```

Für das Argument von SPC gilt dieselbe Begrenzung von 255. Eine doppelte Verwendung von SPC geht natürlich auch, allerdings zählt SPC nicht vom Anfang der Zeile, sondern ab der letzten Cursor-Stelle. Durch Rechnen oder einfach durch Probieren finden wir die Gesamtzahl von 397, das gibt:

```
1000 PRINT SPC (255) SPC (142) TI/60"SEKUNDEN":END
```

Es gibt noch eine dritte Methode, um PRINT-AT zu simulieren.

In die Speicherzelle 214 kann die Zahl einer Zeile hineingePOKEt werden, auf die mit einem nachfolgenden PRINT der Cursor gesetzt wird. Das gleiche gilt für einen Platz in einer Zeile mit der Speicherzelle 211. Versuchen Sie es mit der direkten Eingabe:

```
POKE 214,8:PRINT:POKE 211,4:PRINT"A"
```

Drei Formen für »PRINT AT«

Das druckt den Buchstaben A in die 4. Spalte auf der 9. Zeile. Für unseren Anwendungsfall in Zeile 1000 müssen wir die Zahl 18 nach 214 POKEn, 211 können wir vernachlässigen.

```
1000 POKE 214,18:PRINT:PRINT TI/60"SEKUNDEN":END
```

Alle drei Methoden sind gleichwertig, sowohl in Auswirkung als auch beim Speicherbedarf. Ich bleibe im folgenden bei der 214-Methode.

Zurück zur Version 1 des Testprogramms. Das Programm unterscheidet sich für die beiden Computer nur in der Zeile 40, allerdings auch durch die Laufzeit. Nach RUN erhalten wir 10,48 Sekunden.

Dieses Auffüllen des Bildschirms mit A geht halt recht langsam. Schon beim Zuschauen wird man ungeduldig. Der Teil in diesem Programm, welcher die Laufzeit am nachhaltigsten beeinflußt, ist die 374fache Wiederholung des POKE-Befehls. Bei einem POKE-Befehl ist die Umwandlung der Zahlen aus dem ASCII-Code sehr zeitaufwendig. Hoppla, was heißt denn das schon wieder, sagen Sie jetzt vielleicht.

Jede Zahl, wie zum Beispiel 7680 oder 1024, wird zuerst als vier einzelne ASCII-Codezahlen gespeichert. Wenn das Programm abläuft, werden diese ASCII-Zahlen zuerst in ganze Zahlen, dann in Fließkomma-Zahlen umgewandelt – das für den Fall, daß mit den Zahlen arithmetische Funktionen ausgeführt werden. Schließlich werden sie wieder in eine ganzzahlige POKE-Adresse umgewandelt, und das in unserem Fall 374mal!

Hier können wir einen ersten innerbetrieblichen Verbesserungsvorschlag einreichen. Wenn wir eine so häufig vorkommende Zahl wie die POKE-Adresse in Zeile 40 am Anfang des Programms einer Variablen zuweisen, dann erfolgt die oben genannte Umwandlungssequenz nur einmal, nämlich am Anfang des Programms. Das Programm muß dann 374mal nur den Wert der Variablen im Speicher suchen, und das geht viel schneller. Wollen Sie es sehen? Ändern Sie bitte für diese **Version 2** die Zeilen 30 und 40.

Die Anfangsadresse im Bildschirmspeicher definieren wir als Variable mit dem schönen und zutreffenden Namen »VIDEO«, die des Farbspeichers mit »FARBE«. In Zeile 30 erhalten sie dann die Werte

```
30 Z=0:VIDEO=1024:FARBE=55296
```

Zeile 40 ergibt sich dann eigentlich zwangsläufig:

```
40 POKE VIDEO+Z,1:POKE FARBE+Z,14
```

Alles andere bleibt gleich. Tippen Sie RUN ein. Ergebnis: 7,3 Sekunden. Wir haben also eine Verkürzung von zirka drei Sekunden erzielt, das sind 30 Prozent!

Eine Beschleunigung von 30 Prozent ist gut, aber noch nicht alles, was wir erreichen können. Da die Methode der vordefinierten Variablen so effektiv ist, wollen wir sie auf alle oft verwendeten Zahlen des Programms anwenden. Neben Z, VIDEO und FARBE gibt es noch die 1 für den Buchstaben A und die 14 für die Farbe sowie den Schlußwert 374 der Schleife. Sie wissen schon, wie das geht. Wir ändern folgende Zeilen:

```
30 Z=0: SCHLUSSWERT=374:VIDEO=1024:FARBE=55296:
BUCHSTA=1:DRUCK=14
40 POKE VIDEO+Z, BUCHSTA:POKE FARBE+Z,DRUCK
60 IF Z=SCHLUSSWERT THEN 1000
```

Das ist **Version 3** des Programms, mit einer Laufzeit von 6,15 Sekunden. Das ist wieder eine Verbesserung gegenüber der ersten Version.

Bisher haben wir die Variablen im Sinn einer guten Lesbarkeit mit langen und verständlichen Namen versehen. Aber das kostet natürlich Speicherplatz und auch Geschwindigkeit. Der Grund ist immer derselbe: Bei der Variablen VIDEO sind fünf Zeichen 374mal zu bearbeiten. Wenn wir sie nur VI nennen, ist das erheblich weniger. In **Version 4** des Programms reduzieren wir also alle langen Variablennamen auf zwei Zeichen. Wir wollen mal schauen, was das bringt:

```
30 Z=0: SC=374: VI=1024:FA=55296:BU=1:DR=14
40 POKE VI+Z,BU:POKE FA+Z,DR60 IF Z=SC THEN 1000
```

Ergebnis: 5,63 Sekunden

Gegenüber der Version 1 des Programms haben wir schon eine Verbesserung von 46% erreicht.

Sie wissen sicher, daß der Computer alle Variablennamen immer auf zwei Stellen reduziert, daß aber einstellige Variablen durchaus zugelassen sind. Damit müßte unser Programm eigentlich noch schneller werden. Die folgenden Änderungen für **Version 5** zielen genau darauf:

```
30 S=374: V=1024: F=55296:B=1:D=16
40 POKE V+Z,B:POKE F+Z,D60 IF Z=S THEN 1000
```

Die erzielte Verbesserung ist meßbar, aber nicht gerade überwältigend. Der C 64 braucht nun 5,51 Sekunden. Mit dem bisher Gesagten läßt sich bereits eine erste generelle Regel für schnellere Basic-Programme aufstellen.

Regel 1

* Häufig vorkommende Zahlen, Adressen und Variable, besonders innerhalb von Schleifen, werden am Anfang des Programms vordefiniert.
* Variable, die sich im Lauf des Programms verändern, werden trotzdem vordefiniert, allerdings mit einem unschädlichen Anfangswert (dummy).
* Die Zahl, die am häufigsten vorkommt, wird als erste vordefiniert (damit sie schneller »gefunden« wird).
* Variablennamen sollen möglichst einstellig, aber höchstens zweistellig sein.

Halt! Löschen Sie das Programm noch nicht. Wir liegen noch unter 50 Prozent mit unseren Verbesserungen, und das reicht noch lange nicht. Was können wir am bisherigen Programm noch ändern? Denken Sie mal nach. Wie kann man Buchstaben auf den Bildschirm bringen? Natürlich, mit PRINT statt der POKEs.

In **Version 6** des Programms ersetzen wir die Buchstaben-POKE-Befehle durch eine PRINT-Anweisung, die prinzipiell noch den Vorteil hat, daß keine Farbanweisung nötig ist. Den Buchstaben A wollen wir in dieser Version zunächst einmal mit seiner ASCII-Codezahl 65 angeben, ein Semikolon setzt alle A hintereinander.

```
40 PRINT CHR$(65);
```

Für die Schleife brauchen wir nur die Variablen Z und S:

```
30 Z=0:S=374
```

Alle anderen Zeilen bleiben unverändert.

Nach RUN sehen wir als Ergebnis: 4,05 Sekunden

Das ergibt eine weitere Verbesserung von 1,5 Sekunden. Unser Programmbeispiel wird also durch die Verwendung von PRINT statt POKE stark beschleunigt. Das geht natürlich deswegen besonders gut, weil alle Buchstaben automatisch hintereinander gesetzt werden. Wenn wir jedesmal den Platz mit angeben müßten, wohin gePRINTet werden soll, wäre der Vorteil rasch verspielt.

64er-online.de 64er-online.net

Die Anweisung PRINT CHR$(65) ist zwar gut lesbar, aber wir haben ja vorher gelernt, daß Vordefinieren von Variablen schneller ist. In **Version 7** machen wir das auch mit der PRINT-Variablen.

```
30 Z=0: S=374: A$=CHR$(65)
40 PRINT A$
```

Das Resultat ist wieder beeindruckend: 3,10 Sekunden. Das sind schon 69 Prozent Verbesserung gegenüber der 1. Version.

Aber selbst - oder gerade - jedem Anfänger ist die direkte PRINT-Anweisung mit Gänsefüßen am geläufigsten. Sie braucht auch weniger Speicherplatz und macht sogar ein Vordefinieren unnötig. Wir wollen schauen, ob sie auch schneller ist.

Ändern Sie für **Version 8** die Zeilen 30 und 40:

```
30 Z=0:S=374
40 PRINT "A";
```

Erstaunlicherweise bringt das fast gar nichts, nur 0,07 Sekunden. Die Erklärung liegt darin, daß beide Darstellungen, CHR$(65) und "A" ASCII-Code-Verwender sind. Damit ist der einzige Unterschied zwischen Version 7 und 8 die Anzahl der Zeichen im Programm. Fassen wir zusammen:

Regel 2
* In Schleifen mit aneinandergereihten Druckanweisungen ist PRINT viel schneller als POKE.
* Die PRINT-Variablen sollen entweder vordefiniert oder im Gänsefuß-Modus eingesetzt werden.

In der PRINT-Schleife (Zeile 40) und nachfolgender IF-Abfrage (Zeile 60) gibt es noch zwei Feinheiten. Basic erlaubt uns bei bedingten Sprüngen statt IF .. THEN GOTO .. nur IF .. THEN .. zu schreiben, und das haben wir bisher auch brav gemacht.

Man kann aber auch IF .. GOTO verwenden. Eigentlich ist nicht zu erwarten, daß zwischen den beiden Schreibarten ein Zeitunterschied besteht. Der Fall ist tatsächlich fast akademisch, wie **Version 9** beweist:

```
60 IF Z=S GOTO 1000
```

Laufzeit des C64: 3,0 Sekunden.

Eine andere Änderung bringt auch nur ganz wenig in der Geschwindigkeit, aber sie spart uns eine ganze Zeile und damit Speicherplatz:

```
60 IF Z<>S GOTO 40
70 entfällt
```

Diese **Version 10** gewinnt nur 0,02 Sekunden.

Regel 3
* Bei Schleifen mit Sprunganweisungen ist IF .. GOTO schneller als IF..THEN
* Prüfung auf Ungleichheit (<>) bietet Vorteile, wenn sie eine Zeile erspart.

Für die etwas weiter Fortgeschrittenen unter Ihnen gebe ich hier noch eine weitere Anregung, die wir mit unserem Prüfprogramm nicht testen können.

Die Prüfung mit IF..THEN hängt oft von mehr als einer Bedingung ab. Zum Beispiel kann sie so lauten:

```
100 IF (A=1 AND B=2 AND C=3) THEN 999
110 GOTO 50
```

Zeile 100 prüft jedesmal, ob alle drei Bedingungen erfüllt sind, erst nach dem THEN wird entschieden, ob das Programm auf Zeile 999 springt oder auf 110 weiterläuft. Nehmen wir an, A ist im 20. Durchlauf erfüllt, B im 50. Durchlauf, C aber erst im 300. Durchlauf. Das Programm muß also 300mal alle drei Bedingungen nachprüfen. Wenn wir die Zeile 100 so schreiben:

```
100 IF C=3 THEN IF B=2 THEN IF A=1 THEN 999
```

dann bricht das Programm 300mal die Prüfung nach dem C sofort ab und geht in 110 weiter. B und A werden erst dann zur Prüfung herangezogen, wenn C=3 ist. Es ist wohl klar und einzusehen, daß die zweite Schreibweise schneller ist.

Regel 4
Bei IF..THEN-Prüfungen mit mehreren Bedingungen sollen diese Bedingungen in einzelnen IF..THEN-Prüfungen hintereinander gesetzt werden. Dabei wird die Bedingung an die erste Stelle gesetzt, welche als erste nicht erfüllt ist.

Ich habe mit Absicht bis hierher die 374malige Wiederholung der Schleife mit der Zählvariablen Z hochgezählt und mit IF..THEN beziehungsweise IF..GOTO den Schlußwert abgefragt. Das gab mir die Gelegenheit, die Schnelligkeit dieser Methode ganz auszureizen. Und wir haben auch die ursprüngliche Laufzeit von 10,48 Sekunden auf 2,88 Sekunden reduziert! Jetzt wollen wir noch eine andere Basic-Möglichkeit austesten, die Sie ja alle kennen, nämlich die Schleife mit FOR..TO..NEXT zu programmmieren.

Wir wollen in **Version 11** des immer noch gleichen Programms dazu die Zeilen 30, 50 und 60 ändern:

```
30 FOR Z=1 TO 374
50 NEXT Z
60 entfällt
```

Lassen Sie's laufen. Huiiih! Das pfeift runter! Der C64 braucht nur noch 1,08 Sekunden. Das bringt gegenüber der IF..THEN-Schleife der Version 10 eine ganze Menge, nämlich ungefähr 2 Sekunden oder, auf die Zeit der Version 1 bezogen 89 Prozent.

Das einzige, was wir in der FOR..NEXT-Schleife noch verbessern können, ist die vielgeübte Praxis des Weglassens der Schleifenvariable Z nach dem NEXT-Befehl.

```
50 NEXT
```

Der Geschwindigkeitsgewinn dieser **Version 12** ist nicht sehr groß, nämlich nur etwa 0,1 Sekunden, aber wir wollen die Methode doch in die nächste Regel mit aufnehmen.

Regel 5
Schleifen sollen nicht mit IF..THEN, sondern mit FOR..TO..NEXT gebildet werden. Die Schleifenvariable nach NEXT soll dann weggelassen werden, wenn es nicht zu Verwechslungen mit anderen Schleifen führen kann.

Ich bin fast am Ende meines Beschleunigungslateins. Nur eines bleibt noch, nämlich die bisher hochgehaltene Lesbarkeit des Programms zu opfern. Ich hoffe nämlich, Sie haben bisher meiner Eingangsforderung Folge geleistet und alles schön mit Leerzeichen geschrieben. Das behalten wir zunächst noch bei, im Gegenteil, wir wollen zunächst die Lesbarkeit noch erhöhen und REM-Erläuterungen einfügen. Ich schlage vor, die **Version 13** so auszuschmücken:

```
  10 TI$="000000":REM UHR AUF NULL
  12 REM******************************
  13 REM*                            *
  14 REM*TEST-PROGRAMM*              *
  15 REM*                            *
  16 REM******************************
  20 PRINT CHR$(147);:REM ALLES LOESCHEN
  30 FOR Z=1 TO 374:REM 374 ZEICHEN
  40 PRINT"A":
  50 NEXT
 999 REM ZEIT AUSDRUCKEN
1000 POKE 214,18:PRINT TI/60 "SEKUNDEN":END
```

Sieht gut aus, nicht wahr?

Aber leider, REM-Erläuterungen kosten Zeit. Wir sind um 0,2 Sekunden langsamer geworden.

Wir schmeißen deshalb alle REMs wieder raus und haben damit wieder Version 12. Jetzt aber gehen wir einen Schritt in der anderen Richtung weiter und entfernen alle Leerstellen und Abstände. Mit dieser **Version 14** will ich Ihnen zeigen, daß das auch einen Einfluß auf die Laufgeschwindigkeit hat.

```
  10 TI$="000000"
  20 PRINTCHR$(147);
  30 FORZ=1TO374
  40 PRINT"A";
  50 NEXT
1000 POKE214,18:PRINT:PRINTTI/60"SEKUNDEN":END
```

Das Ergebnis ist 0,98 Sekunden.

In **Version 15** treiben wir die Schrumpfung ins Extrem, indem wir das Programm im Prinzip unverändert, aber mit

einem Minimum an Zeilen schreiben, also möglichst viele Befehle in eine Zeile packen.

Sie wissen, die maximale Zeilenlänge beträgt 80 Zeichen beim C64. Unser Programm können wir sogar in einer einzigen Zeile unterbringen – fast unglaublich, aber es geht. Sie müssen allerdings alle Abkürzungsmöglichkeiten ausschöpfen, die das Commodore-System bietet. Im Anhang der Commodore-Handbücher finden Sie die Liste aller Abkürzungen beim Eintippen: C und geSHIFTetes H für CHR$, ? für PRINT und so weiter. Im nachfolgenden Ausdruck ist das natürlich nicht zu sehen, weil der Befehl LIST die Abkürzungen nicht berücksichtigt. So kommen auch mehr als 80 Zeichen in eine Zeile des Listings.

```
10 TI$="000000":PRINTCHR$(147);:FORZ=1TO374:
PRINT"A";:NEXT:POKE214,18:PRINT:PRINT TI/60
"SEKUNDEN":END
```

Und siehe da, diese »Kurzform« des Programms ist auch die allerschnellste Version. Der C64 braucht 0,93 Sekunden. Diese letzte Beschleunigung wird dadurch erreicht, daß das Betriebssystem des Computers nur einmal einen Zeilenanfang und Zeilenende suchen und erkennen muß, statt sechsmal in der Version 14.

Das Ausnützen der vollen Kapazität einer Zeile bringt also nicht nur den Vorteil eines kleineren Speicherbedarfs, sondern auch Zeitgewinn.

Regel 6
* Programme ohne REM-Erläuterungen und ohne Leerstellen zwischen den Zeichen laufen schneller.
* Zur Reduzierung der Zeilenzahl sollen möglichst viele Befehle in eine Zeile geschrieben werden.

»Einzeiler« können auch Spaß und Herausforderung zugleich sein. Man sollte eigentlich annehmen, daß mit einer Zeile nicht viel anzufangen sei. Weit gefehlt!

Einzeilige Programme

Ich schreibe die Einzeiler unten lesbar, das heißt mit Leerstellen. Sie müssen die Programme aber wieder mit allen Abkürzungstricks schreiben, sonst geht's schief.

Laufzeiten der Programmversionen für »Buchstaben auf dem Bildschirm«

Version	Programmier-Methode	Laufzeit (Sek.)
1	Buchstaben POKEn,Zählschleife mit IF ... THEN	10,48
2	POKE-Adressen vordefinieren	7,30
3	alle Variablen vordefinieren	6,15
4	Variablen mit 2 Buchstaben	5,63
5	Variable mit 1 Buchstaben	5,51
6	PRINT CHR $ statt POKE	4,05
7	CHR $ vordefinieren	3,10
8	PRINT "A"	3,03
9	Schleife mit IF-GOTO statt IF-THEN	3,00
10	IF Z<>S statt IF Z = S	2,98
11	Schleife mit FOR-NEXT	1,08
12	NEXT ohne Zählvariable	1,00
13	Listing mit REMs	1,20
14	Listings ohne REM, ohne Abstände	0,98
15	Alles in einer Zeile	0,93
16	Maschinensprache	0,066

Von A. Boyd (Manchester) stammt ein Primzahlen-Erzeuger, der für die obere Grenze von 65000 viele Stunden braucht.

```
1 FOR N=1 TO 65000:F=0FOR J=2 TO N-1:F=F+((N-J*INT
(N/J))=0):NEXT:X=-(F=0):PRINT RIGHT$(STR$(X*N),6*X):
NEXT
```

Ein anderer Einzeiler wurde von A.M. Simmelt (Sheffield) geschrieben zur Konvertierung von Dezimal- in Dualzahlen.

```
1 INPUT A:FOR I=14 TO 0 STEP-1:Z=A AND 2↑I:A=A-Z:Z
=Z/2↑I:Z$=RIGHT$(STR$(Z),1):PRINT Z$;:NEXT:
GOTO1
```

Lassen wir's gut sein mit diesem Programmsport und kehren wir zurück zu einer abschließenden Betrachtung der Zeitgewinne.

Wir haben in Version 1 mit 10,48 Sekunden begonnen. Diese Laufzeit wurde ohne Änderung des Programmresultats stetig verkürzt, bis wir schließlich in Version 15 bei 0,93 Sekunden gelandet sind. Ich nenne diese Beschleunigung um 90 Prozent schlicht und einfach spektakulär.

Mehr allerdings kann ich nicht herausholen, es sei denn – na ja, eigentlich habe ich am Anfang ganz laut »Basic« gesagt.

Aber ich kann doch nicht widerstehen und Sie scharf machen auf ultima velocitas – zu deutsch Maschinensprache.

Es soll Ihnen aber kein Maschinensprache-Kurs zugemutet werden. Doch ein Programm in Maschinensprache besteht genauso aus Befehlen, Adressen und Variablen wie ein Basic-Programm, nur sind sie in einem speziellen Zahlencode geschrieben. Dieser Zahlencode muß in den Arbeitsspeicher geladen werden. Die für uns einfachste Möglichkeit besteht darin, die Zahlen in den Speicher hineinzuPOKEn. Damit wir aber nicht unmäßig viele POKE-Befehle schreiben müssen, legen wir alle Code-Zahlen hinter DATA-Befehle und holen sie dann mit READ in eine einzige POKE-Schleife. Ich sage das deswegen, weil dieses Einlesen natürlich nicht zu dem Testprogramm gehören darf, dessen Laufzeit wir messen wollen. Das Testprogramm selbst sitzt zwischen den drei Zeilen der »Stoppuhr«. Das heißt, genauer gesagt sitzt das Programm in den Speicherzellen, in die wir es hineinPOKEn. Aber zwischen der Stoppuhr rufen wir es auf, der dem RUN entsprechende Befehl bei Maschinensprache heißt SYS.

Wie Sie gleich noch sehen werden, fängt unser Testprogramm ab Speicherzelle 7168 an. Das Ganze sieht dann so aus:

```
10 TI$="000000"
20 PRINT CHR$ (147)
30 SYS 7168
1000 POKE 214,18:PRINT:PRINT TI/60 "SEKUNDEN":END
```

Ab Zeile 2000 setzen wir jetzt das Programm, welches uns das Maschinenprogramm einliest. Um mit dem Einlesen zu beginnen, setzen wir noch eine Umleitung vor das Meßprogramm: `5 GOTO 2000`

In Zeile 2000 löschen wir den Bildschirm. Zeile 2010 und 2020 und 2030 liest die Codezahlen, die von Zeile 2050 bis 2090 stehen, und POKEt sie in die Speicherplätze 7168 bis 7200. Sobald die Zahlen eingelesen sind, können Sie das Meßprogramm mit dem Befehl GOTO 10 (direkt eingetippt) starten.

Im Abdruck unten wird das etwas eleganter gemacht. Zuerst meldet das Programm das Ende des Einlesens (Zeile 2100 und 2101). Dann kommt die Anweisung, wie das Meßprogramm zu starten ist, nämlich durch Drücken irgendeiner Taste, die durch eine GET-Schleife abgefragt wird. Wenn eine Taste gedrückt wird, springt das Programm auf Zeile 10 (das geschieht in Zeile 2160).

```
   5 GOTO 2000
  10 TI$="000000"
  20 PRINT CHR$(147)
  30 SYS 7168
1000 POKE 214,18:PRINT:PRINT TI/60 "SEKUNDEN":END
2000 PRINT CHR$(147)
2010 FOR A=7168 TO 7200
2020 READ B
2030 POKE A,B
2040 NEXT
2050 DATA 162,0,169,1,157
2060 DATA 0,4,169,14,157,0,216
2070 DATA 232,224,0,208,241,169,1,157
2080 DATA 254,4,169,14,157,254,216
2090 DATA 232,224,120,208,241,96
2100 PRINT"DAS MASCHINENPROGRAMM"
2110 PRINT"IST JETZT EINGELESEN.":PRINT
2120 PRINT"ZUM STARTENDES PRO-"
2130 PRINT"GRAMMS" CHR$(18)"TASTE"CHR$(146)
"DRUECKEN"
2140 GET A$
2150 IF A$="" THEN 2140
2160 GOTO 10
```

So, inzwischen haben Sie sicher Ihre Überraschung gehabt! 0,066 Sekunden Laufzeit. Ich hoffe, daß ich Sie mit dem Virus der Maschinensprache infiziert habe.

Wir wollen im folgenden ein paar arithmetische Funktionen untersuchen und beschleunigen. Als erste nehmen wir uns in **Version 17** die Multiplikation vor. Die Messung der Laufzeit erfolgt auf dieselbe Weise wie bei allen Programmen vorher auch. Deshalb bleiben die Zeilen 10, 20 und 1000 gleich. Die Multiplikation selbst soll 300mal ausgeführt werden (Zeile 30). Dann wird das Ergebnis gedruckt (Zeile 60).

```
30 FOR Z=1 TO 300
50 NEXT
60 PRINT A
```

Als Multiplikation nehmen wir den Extremfall einer kurzen Zahl multipliziert mit einer langen.

```
40 A=3*0,123456789
```

Nach RUN bleibt der Bildschirm zuerst leer, bis dann nach 11,85 (14, 15) Sekunden das Ergebnis der Multiplikation und die Laufzeit ausgedruckt wird.

In **Version 18** vertauschen wir die beiden Zahlen, die in Zeile 40 multipliziert werden.

```
40 A=0.123456789*3
```

Diese einfache Manipulation bringt natürlich nach Adam Riese dasselbe Resultat wie vorher, aber die Laufzeit ist kürzer. Wir gewinnen 0,44 Sekunden. Dieser Gewinn ist nicht überwältigend, aber überraschend. Aber denken Sie nach!

Wie ist das, wenn Sie so eine Multiplikation auf dem Papier durchführen? Da ist die Rechnung im zweiten Fall auch einfacher. Der Computer hat genau dasselbe Problem.

In **Version 19** nützen wir noch eine kleine Eigenheit der Commodore-Computer aus, die auf ihre amerikanische Herkunft zurückzuführen ist. Bei den Angelsachsen ist es nämlich erlaubt, eine Null vor dem Dezimalpunkt wegzulassen. Beim Computer dürfen wir das auch. Obwohl das mit der Multiplikation direkt nichts zu tun hat, bietet sie uns doch eine gute Gelegenheit, die Einsparung durch das Weglassen der Null auch zeitlich zu messen. Also, Zeile 40 sieht jetzt so aus:

```
40 A=.123456789*3
```

Das bringt nicht sehr viel, 0,20 Sekunden. Aber Kleinvieh macht auch Mist.

Eine ähnliche Verbesserung, die wir hier nicht ausprobieren, wird erzielt durch den Ersatz einer alleinstehenden Null durch einen Punkt, zum Beispiel:

statt IF X=0 THEN –
jetzt IF x=. THEN –

Eine gewaltige Beschleunigung erfährt das Multiplikationsbeispiel, wenn wir die Regel 1 anwenden und die Variablen vordefinieren.

Regel 7
* Bei einer Multiplikation soll die längere Zahl vor der kürzeren stehen (langer Multiplikant, kurzer Multiplikator).
* Eine einzelne Null wird durch einen Punkt ersetzt, eine Null vor dem Dezimalpunkt wird weggelassen.

In **Version 20** ersetzen wir in Zeile 40 beide Zahlen durch Buchstaben, die wir in einer neuen Zeile 25 diese Werte zuweisen.

```
25 B=.123456789:C=3
40 A=B*C
```

Dieser Lauf bleibt nach 1,48 Sekunden stehen, das heißt wir gewinnen 12,23 Sekunden. Also bitte Regel 1 unbedingt beachten!

Eine andere betrachtenswerte arithmetische Funktion ist das »Potenzieren« (Quadrat-/Kubikzahlen), ausgelöst durch das Zeichen ↑. **Version 21** erzielen wir durch Löschen der Zeile 25 und Abänderung der Zeile 40:

```
40 A= 4 ↑ 3
```

»Vier hoch drei« ergibt 64 und braucht 10,53 Sekunden.

In **Version 22** wollen wir sehen, ob vordefinierte Variable auch so einschlagen wie bei der Multiplikation.

```
25 B=4:C=3
40 A=B↑C
```

Man kann sich doch auf nichts verlassen! Diesmal sind wir nur um 0,22 Sekunden schneller. Wir dürfen aber nicht aufgeben. **Version 23** macht alles wieder wett, und zwar durch den simplen Trick, daß wir das Potenzieren in seine Grundelemente zerlegen.

Sie wissen doch: 4 hoch 3 (4↑3) ist dasselbe wie »4 zweimal mit sich selbst multipliziert« (4*4*4).

```
25 B=4 (C entfällt)
40 A=B*B*B
```

Ja, da schauen Sie, gell? Das Programm braucht nur 1,68, also 8,31 Sekunden weniger.

Regel 8
Die Funktion Potenzieren (↑) soll durch Mehrfach-Multiplikation ersetzt werden.

Als letztes Objekt möchte ich oft aufgerufene Unterprogramme messen. Wir erreichen das ganz einfach dadurch, daß wir das letzte Programm (Version 23) abändern. So erhalten wir **Version 24:** Die Definition der Variablen (Zeile 25) und die Multiplikation (Zeile 40) verbannen wir als Unterprogramm an das Ende des Programms und springen innerhalb der 300fachen Schleife mit GOTO darauf.

```
25 löschen;
30 FOR Z=1 TO 300;
40 GOTO 40000
```

Alles andere bleibt, aber neu kommt dazu: `40000 B=4;`

```
50000 A=B*B*B;
60000 GOTO 50
```

Es ist nicht weiter erstaunlich, daß dieser Umbau diese Version 24 gegenüber Version 23 verlangsamt. Aber merken Sie sich die Laufzeit, 3,28 Sekunden. Als nächstes ersetzen wir die beiden GOTO-Zeilen durch GOSUB-RETURN.

```
40 GOSUB 40000
60000 RETURN
```

Diese **Version 25** spart uns 0,15 Sekunden. GOSUB ist schneller als GOTO! Sie haben vielleicht schon gelesen, daß oft gebrauchte Unterprogramme am Anfang eines Programms stehen sollen. Den Grund dafür will ich Ihnen mit den nächsten zwei Versionen vorführen.

Version 26 macht das zunächst für die GOTO-Version. Wir bauen sie auf der Version 24 auf, mit folgenden Änderungen:

Die Zeitmessung lassen wir wie gehabt in den Zeilen 10, 20 und 1000, die Schleife und den Ausdruck des Resultats in den Zeilen 30, 50 und 60.

Nur beim Unterprogramm streichen wir alle Nullen der Zeilennummern, so daß es jetzt in den Zeilen 4, 5 und 6 steht. Um zu vermeiden, daß das Programm gleich mit dem Unterprogramm beginnt, fügen wir davor (Zeile 3) noch eine Umleitung ein, die sofort auf der Zeile 10 weitermacht. Schließlich brauchen wir noch den Sprung in das Unterprogramm, den wir in die Zeile 33 setzten. Das Ganze sieht jetzt so aus:

```
3 GOTO 10
4 B=4
5 A=B*B*B
6 GOTO 50
10 TI$="000000"
20 PRINT CHR$(147)
30 FOR Z=1 TO 300
33 GOTO 4
50 NEXT
60 PRINT A
1000 POKE 214,18:PRINT:PRINT TI/60 "SEKUNDEN":END
```

Nach RUN erhalten wir 3,1 Sekunden. Gegenüber Version 24, unserem Vergleichsobjekt, sparen wir 0,18 Sekunden.

Dasselbe passiert, wenn wir in der **Version 27** die GOTOs mit GOSUB-RETURN ersetzen.

```
6 RETURN
33 GOSUB 4
```

Gegenüber der anderen GOSUB-Version (Version 25) sparen wir 0,17 Sekunden.

Regel 9
* Der Aufruf von Unterprogrammen mit GOSUB ist schneller als mit GOTO.
* Häufig gebrauchte Unterprogramme gehören ganz an den Anfang eines Programms. Sie müssen dann allerdings zuerst mit einem GOTO umgangen werden.

Ich bin überzeugt, daß in Basic noch mehr spektakuläre Zeitgewinne stecken. Falls Sie eine Regel 10 oder noch mehr entdecken, ermuntere ich Sie um Mitteilung.

Wenn Sie Fragen haben, können Sie mich mit einer Leserzuschrift ansprechen. (Dr. Helmuth Hauck/aa/hm)

Version	Programmier-Methode	Laufzeit (Sek.)
17	Multiplikation, lang x kurz	14,15
18	Multiplikation, kurz x lang	13,71
19	Null weglassen	13,51
20	Variable vordefinieren	1,48
21	Potenzieren (4 hoch 3) mit ↑	10,53
22	Variable vordefinieren	10,31
23	4 x 4 x 4 statt 4↑3	2,01

Laufzeiten der Versionen für arithmetische Funktionen

Version	Programmier-Methode	Laufzeit (Sek.)
24	Unterprogramm am Ende, Sprung mit GOTO-GOTO	3,28
25	Unterprogramm am Ende, Sprung mit GOSUB-RETURN	3,13
26	Unterprogramm am Anfang, Sprung mit GOTO-GOTO	3,10
27	Unterprogramm am Anfang, Sprung mit GOSUB-RETURN	2,96

Laufzeiten der Versionen für Unterprogramme

Debugging – Fehlersuche in Basic-Programmen

Diese Situation kennt wohl jeder Besitzer eines Heimcomputers zur Genüge: Da tippt man in stundenlanger Arbeit ein ellenlanges Listing ein, lehnt sich nach dem letzten »RETURN« einen Augenblick erleichtert zurück, gibt das magische Wort »RUN« ein – und natürlich läuft das Programm nicht so, wie es eigentlich sollte.

Das Spektrum der möglichen Ereignisse reicht in einem solchen Fall vom simplen »SYNTAX ERROR« bis zum völligen Absturz des Programms. Solange der Computer noch brav seine Fehlermeldungen ausgibt, hat man ja noch Glück gehabt. Kritisch wird die Situation dann, wenn auf dem Bildschirm ein eigenartiges Gemisch undefinierbarer Zeichen erscheint und sich der Computer weder durch Betätigen aller erreichbaren Tasten noch durch gutes Zureden wieder auf den Boden der Tatsachen zurückholen läßt. Wenn man beim Eintippen eines Programms einmal an diesem Punkt angelangt ist, wird es Zeit, sich die erste Regel gut einzuprägen: Jedes Programm sollte vor dem Start unbedingt mit »SAVE« gesichert werden.

Mit dem Retten des mühevoll erstellten Programms allein ist es allerdings noch nicht getan, es muß effektiv noch etwas gegen die im Programm enthaltenen Fehler unternommen werden. Diesen Vorgang bezeichnet man auch als »Debugging«. Das Wort ist von der englischen Bezeichnung »Bug« abgeleitet und bedeutet eigentlich »entwanzen«, wobei mit den »Wanzen« die Fehler gemeint sind, die sich überall im Programm verstecken. In der amerikanischen Umgangssprache hat sich das Wort ganz allgemein für das Suchen versteckter Fehler eingebürgert.

Ganz grob kann man zwischen zwei Arten von Fehlern unterscheiden. Einerseits gibt es die logischen Fehler, die mit schöner Regelmäßigkeit in der Entwicklungsphase eines Programms auftauchen, weil man dem Computer noch nicht genau genug gesagt hat, was er denn nun eigentlich machen soll. Diese Art von Fehlern erkennt man zumeist daran, daß das Programm anstandslos läuft, aber nicht immer die gewünschten Ergebnisse produziert. Die zweite Art von Fehlern ist wesentlich profanerer Natur und tritt praktisch jedesmal dann auf, wenn man ein Programm von einem fremden Listing oder auch von den eigenen Aufzeichnungen abtippt: Es handelt sich dann zumeist um schlichte Tippfehler oder um Fehler, die auf schlechter Lesbarkeit der Vorlage beruhen. Wir wollen uns im folgenden nur mit der zweiten Art von Fehlern beschäftigen.

Wie geht man nun zweckmäßig vor, um alle Fehler zu finden, ohne das gesamte Programm von Anfang bis Ende mit der Vorlage vergleichen zu müssen? Nun, eine allgemeingül-

tige Methode, die für alle Arten von Programmen anwendbar wäre, gibt es leider nicht. Dennoch erscheint ein gewisses systematisches Vorgehen durchaus angebracht.

Der Computer hilft bei der Fehlersuche

Zunächst sollten wir uns darüber klarwerden, inwieweit uns der Computer selbst bei der Suche nach Fehlern helfen kann. Als erstes kommen einem dabei natürlich die Fehlermeldungen in den Sinn, die beim Commodore-Basic ja erfreulicherweise im Klartext erfolgen und recht vielfältig sind. Wer mit den englischen Bezeichnungen nicht sofort etwas anfangen kann, hat die Möglichkeit, die deutschen Erläuterungen dazu im Handbuch nachzuschlagen.

Was aber soll man tun, wenn der Computer gar keine Fehlermeldung ausgibt, sondern sich nach »RUN« einfach sang- und klanglos verabschiedet und auf keine Tasten mehr reagiert?

Nun, für solche Fälle bietet Basic zwei spezielle Befehle, die man immer dann in nicht zu geringem Umfang einsetzen sollte, wenn man nicht genau weiß, wo denn nun der Fehler steckt. Gemeint sind die Basic-Befehle STOP und CONT. Wenn der Computer beim Abarbeiten des Programms auf den Befehl STOP stößt, unterbricht er die Programmausführung und gibt eine Meldung »BREAK IN nnn« aus, wobei nnn die Zeilennummer ist, in der er die STOP-Anweisung gefunden hat. Alle Variablen und auch der Stackpointer bleiben dabei erhalten, so daß die STOP-Anweisung auch in Unterprogrammen und innerhalb von FOR-NEXT-Schleifen auftreten kann.

Nach einem solchermaßen erzwungenen Programmstopp kann man sich im Direktmodus mit dem PRINT-Befehl über die Werte wichtiger Variablen informieren und sogar mit LIST einzelne Programmteile anschauen. Danach gibt man den CONT-Befehl und das Programm wird ganz normal fortgesetzt. Wenn es zu Testzwecken notwendig erscheint, kann man während eines Stopps auch im Direktmodus Variablenwerte verändern oder FOR-NEXT-Schleifen verwenden. Allerdings dürfen weder neue Programmzeilen eingegeben noch alte gelöscht oder verändert werden, da dadurch gleichzeitig alle Variablen gelöscht werden und CONT danach nicht mehr möglich ist.

Es ist empfehlenswert, an kritischen Stellen im Programm STOP-Befehle einzufügen, um dadurch den Programmlauf verfolgen zu können und den Fehler immer mehr einzugrenzen. Kritische Stellen sind generell und ohne Ausnahme alle SYS- und USR-Aufrufe, desgleichen alle POKE-Befehle, über deren Bedeutung man sich nicht hundertprozentig im klaren ist. Im Zweifelsfalle sollte man auch nach jedem GOSUB im Programm zunächst einen STOP-Befehl einbauen, um sicherzugehen, daß das Unterprogramm auch wieder auf normalem Wege verlassen wird.

Jedesmal, wenn man die Harmlosigkeit zum Beispiel eines SYS-Befehls durch davor und danach plazierte STOP-Befehle festgestellt hat, kann man die STOPs natürlich wieder entfernen, um einen flüssigeren Programmablauf zu erreichen. Es empfiehlt sich, alle eingefügten STOP-Befehle auf einem Zettel zu notieren, um die Übersicht zu behalten. Schließlich dienen diese Befehle nur der Fehlersuche und müssen irgendwann einmal alle wieder entfernt werden.

In vielen Fällen kann man STOP-Befehle durch einfache PRINT-Anweisungen ersetzen. Das hat den Vorteil, daß keine Programmunterbrechung stattfindet und man nicht jedesmal »CONT« eintippen muß. Außerdem kann man in PRINT-Anweisungen auch zusätzliche Informationen geben, zum Beispiel Variablenwerte ausdrucken oder direkt auf ein spezielles Problem aufmerksam machen. Diese PRINT-Anweisungen sollten aber in irgendeiner Weise von den normalen Bildschirmausgaben unterschieden sein. Zum Beispiel kann man jede PRINT-Anweisung zur Fehlersuche mit fünf Sternchen oder fünf Pluszeichen beginnen lassen.

Will man sich mehrere Variable während des Programmlaufs ausdrucken lassen, sind kleine Unterprogramme recht hilfreich, die in einen freien Zeilenbereich geschrieben werden und die alle benötigten Ausgaben durchführen. Am Ende solcher Unterprogramme sollte eine GET-Schleife stehen, die das Programm auf Tastendruck weiterlaufen läßt. Statt langer PRINT-Listen braucht man so nur einen GOSUB-Aufruf überall dort im Programm einzufügen, wo dies sinnvoll erscheint.

Das Arbeiten mit STOP und CONT mag manchem Computerneuling etwas ungewohnt erscheinen, aber es ist jedenfalls ein recht sicheres Mittel, einem immer wieder abstürzenden Programm auf die Schliche zu kommen.

Wenn der C 64 nur noch »Bahnhof« versteht...

Die häufigste Fehlermeldung ist sicherlich der ungeliebte »SYNTAX ERROR«. Böse Zungen behaupten allerdings, beim VC 20 wäre es der »OUT OF MEMORY ERROR«. Wie dem auch sei, solange der Computer nur Syntax-Fehler meldet, kann man noch von Glück reden. Es handelt sich dabei meistens um einfache Tippfehler, die nach Auflisten der entsprechenden Zeile leicht zu finden und zu korrigieren sind. Beliebte Fehler sind zum Beispiel fehlende oder überzählige Klammern und die Verwechselung ähnlicher Zeichen wie zum Beispiel »0« und »O«, »1« und »l« oder »8« und »B«. Sehr häufig ist auch die Verwechslung von Punkt und Komma, was sich besonders in DATA-Zeilen verhängnisvoll auswirken kann, wie wir nachher noch sehen werden. Wenn Sie also irgendwo einen »SYNTAX ERROR« gemeldet bekommen und in der fraglichen Zeile auf Anhieb keinen Fehler finden, dann gehen Sie zuerst die vorhin genannten Punkte durch.

Eine gute Hilfe ist es, mit dem Cursor die fehlerhafte Zeile Zeichen für Zeichen abzufahren und dabei mit der Vorlage zu vergleichen. Kommen in der Fehlerzeile viele Klammern vor, dann empfiehlt sich häufig das Anlegen zweier Strichlisten für öffnende und schließende Klammern. Die Anzahl muß innerhalb jeder Basic-Anweisung übereinstimmen. Aber bitte keine Klammern mitzählen, die in Anführungszeichen stehen, diese haben mit der Syntax nichts zu tun.

Ab und zu kann es vorkommen, daß man bei aller Sorgfalt einen Syntaxfehler nicht findet, wie zum Beispiel in der folgenden Basic-Zeile:

```
10 OPEN 1,4:PRINT#1,"HALLO":CLOSE 1
```

Wenn der Computer hier dennoch einen Syntaxfehler meldet, dann kann das nur eine Ursache haben: Bei der Eingabe dieser Zeile wurden die Basic-Befehle in der bekannten Art und Weise abgekürzt, der zweite Befehl dabei aber als »?#1« eingegeben, was beim Auflisten wieder zu »PRINT#1« wird. Leider sind PRINT und PRINT# zwei völlig verschiedene Befehle, genauso wie INPUT und INPUT# oder GET und GET#. Also bitte bei der abgekürzten Eingabe von »PRINT#« unbedingt die im Handbuch angegebene Form (»P« Shift-»R«) statt »?#« verwenden. Die normale PRINT-Routine weiß nämlich mit dem nachfolgenden »#« nichts anzufangen und es kommt zu einer Fehlermeldung.

Eine ähnliche Situation kann sehr leicht bei Verwendung langer Variablennamen auftreten. In einem Spielprogramm »Schiffe versenken« kann zum Beispiel die folgende Zeile auftreten:

64er-online.de

```
10 ANZAHLSCHIFFE = 12
```

In dieser Zeile wird unweigerlich ein »SYNTAX ERROR« auftreten, weil der Computer innerhalb des Variablennamens »ANZAHLSCHIFFE« das Basic-Schlüsselwort »IF« entdeckt und sich bei aller Anstrengung nicht erklären kann, was eine IF-Abfrage an dieser Stelle soll. Trotz aller Vorteile für die Übersichtlichkeit eines Programmes sei daher an dieser Stelle von der Benutzung langer Variablennamen abgeraten.

Was tun bei »OUT OF DATA«?

Eine andere häufig auftretende Fehlermeldung ist vor allem bei Anfängern gefürchtet, nämlich der »OUT OF DATA ERROR«. Gefürchtet ist dieser Fehler vor allem deswegen, weil die Zeilennummer, die der Computer zu dieser Fehlermeldung ausgibt, in den allermeisten Fällen keinen Hinweis darauf gibt, an welcher Stelle denn nun ein Fehler vorliegt. Listet man nämlich die fehlerhafte Zeile am Bildschirm auf, so findet man dort nur den READ-Befehl, für den keine DATAs mehr vorhanden waren. Einen näheren Hinweis erhält man durch

```
? PEEK(63) + 256 * PEEK(64)
```

Dieser Befehl listet die Zeilennummer der zuletzt gelesenen DATA-Zeile. Ein typisches, wenn auch stark vereinfachtes Beispiel für das Auftreten von Fehlern im Zusammenhang mit DATA-Anweisungen, ist in Listing 1 gegeben. In den Zeilen 100 bis 170 wird eine Prüfsumme über den ersten DATA-Block gebildet und nur dann, wenn diese Prüfsumme in Ordnung ist, wird in den nächsten Programmteil verzweigt, wo aus dem zweiten DATA-Block Zahlen gelesen und an den Anfang des Bildschirms gePOKEt werden und dort das Wort »COMMODORE« bilden sollen.

Das Programm enthält nun einige Fehler in den DATA-Zeilen, die wir gemeinsam herausfinden wollen. Stellen wir uns einfach vor, wir hätten das Programm in Listing 1 aus einer Zeitschrift abgetippt und dabei einige Fehler in den DATA-Zeilen fabriziert. In Listing 2 sind zum Vergleich noch einmal die entsprechenden DATA-Zeilen des »Original«-Listings abgedruckt. Die Fehlersuche scheint somit recht einfach: Man vergleicht die paar DATAs in beiden Listings und wird dann schon den Fehler finden. Bei diesem kurzen Testprogramm stimmt das natürlich auch. Aber stellen wir uns doch einmal vor, daß die beiden DATA-Blöcke insgesamt vielleicht über drei volle Listing-Seiten gehen und nicht nur über drei Zeilen wie in unserem Beispiel. Dann lohnt es sich nämlich mit Sicherheit schon, wenn man etwas systematischer an die Fehlersuche herangeht.

Zuerst starten wir unser Programm nach Listing 1 einmal ganz arglos mit RUN, nachdem wir es vorher auf Kassette oder Diskette gespeichert hatten. Der Programmverlauf ist zu Anfang ganz wie erwartet: Der Bildschirm wird gelöscht, es erscheint die Meldung »S = 270« und darunter »OK«, dann jedoch erscheinen am oberen Bildschirmrand statt eines längeren Wortes nur die beiden Zeichen »%« und »C« und das Programm bricht mit der Meldung »? ILLEGAL QUANTITY ERROR IN 230« ab. Was ist hier geschehen?

Wenn wir uns Zeile 230 einmal auflisten lassen, dann sehen wir

```
230 POKE B+I,X
```

Da wir wissen, daß nur Zahlen zwischen 0 und 255 gePOKEt werden können, vermuten wir den Fehler beim Wert der Variablen X. Um unsere Vermutung zu bestätigen, fragen wir den Computer doch einmal ganz einfach nach dem Wert von X, indem wir eintippen

```
PRINT X
```

und danach die RETURN-Taste betätigen. Wir erhalten als Antwort den Wert 1513, der tatsächlich zu groß ist, um in eine Speicherzelle zu passen. Wir vergleichen den zweiten DATA-Block mit dem Original (Listing 2) und stellen fest, daß wir bei der Eingabe das Komma zwischen den beiden Zahlen 15 und 13 in Zeile 360 vergessen haben. Das ändern wir, indem wir das Komma nachträglich einfügen.

Dank dieses schnellen Erfolges bessert sich unsere Stimmung um einiges, was sich jedoch nach dem nächsten RUN sehr schnell wieder ändert. Zwar erscheint am Bildschirm zunächst ganz ordentlich die Prüfsumme des ersten DATA-Blocks und das dazugehörige »OK«, aber am oberen Bildschirmrand stimmt einiges noch nicht: Man liest dort die Zeichenfolge »@COMMOORE« statt »COMMODORE«.

Kein Verlaß auf Prüfsummen

Auf den ersten Blick würde man vielleicht vermuten, daß der Fehler nur im zweiten DATA-Block stehen kann, weil das Lesen des ersten Blocks keine Fehlermeldung erzeugt und sogar die Prüfsummenbildung stimmt. Diese Überlegung ist aber nicht ganz schlüssig. Denn durch Bildung einer einfachen Prüfsumme werden Vertauschungsfehler und überflüssige oder fehlende Nullen nicht erkannt. Die fünf DATA-Zeilen in Listing 3 ergeben zum Beispiel alle die gleiche Prüfsumme.

Man sollte sich also nie blindlings auf Prüfsummen verlassen. Sie sind zwar oft nützlich, um Fehler in DATA-Zeilen festzustellen, man darf aber aus der Richtigkeit der Prüfsummenprobe niemals auf die Abwesenheit von Fehlern schließen. Außerdem taucht eine weitere Schwierigkeit auf: Wenn man nur eine globale Prüfsumme über alle DATAs bildet, dann kann man zwar unter Umständen einen Fehler nachweisen, weiß aber immer noch nicht, wo er steckt. Da muß man dann schon zu anderen Mitteln greifen.

Um den Fehler aufzuspüren, können wir dem Computer einen großen Teil der Arbeit überlassen. Als erstes wollen wir feststellen, in welchem der beiden DATA-Blöcke der Fehler liegen könnte. Dazu veranlassen wir den Computer einfach, nur den ersten DATA-Block zu lesen, indem wir eine STOP-Anweisung hinter die erste FOR-NEXT-Schleife plazieren. Wir fügen also folgende Zeile ins Programm ein:

```
145 STOP
```

Wenn wir das Programm jetzt starten, erhalten wir die Meldung »BREAK IN 145«, die von unserem STOP-Befehl herrührt. Da aber das Programm bis Zeile 145 durchlaufen wurde, muß der erste DATA-Block an dieser Stelle vollständig gelesen worden sein. Die Variable X enthält natürlich immer noch den zuletzt gelesenen DATA-Wert. Wenn dieser Programmteil richtig gearbeitet hat, dann müßte X jetzt den Wert Null haben, denn dies ist ja gerade der letzte DATA-Wert aus Block 1, wie man anhand von Listing 1 oder 2 unschwer erkennen kann. Das können wir einfach nachprüfen, indem wir den Computer nach dem Wert von X fragen:

```
PRINT X
```

Zu unserem Erstaunen ist die Antwort aber nicht 0, sondern 12. Wir werfen wieder einen Blick auf Listing 1 und stellen fest, daß die Zahl 12 die vorletzte Zahl im ersten DATA-Block ist. Offenbar wurde eine Zahl zuwenig gelesen ! Es ist nun verlockend, einfach den Endwert der ersten FOR-NEXT-Schleife um eins zu erhöhen, um alle Werte des ersten Blocks zu lesen. Doch halt, hier ist Vorsicht geboten. Viel wahrscheinlicher als ein Fehler in einer FOR-NEXT-Schleife

ist ein Fehler innerhlb der DATA-Zeilen. Bei so vielen Zahleneingaben kann man sich schließlich leicht mal vertippen. Betrachten wir das Problem also einmal von der anderen Seite. Wenn die Anzahl der gelesenen X-Werte stimmt, das Programm aber trotzdem nur bis zum vorletzten DATA-Wert kommt, dann enthält Block 1 vielleicht einen DATA-Wert zuviel. Wir wollen also die DATAs in Block 1 ganz gezielt überprüfen. Dazu schreiben wir in einen freien Zeilenbereich, zum Beispiel ab Zeile 1000, das folgende kleine Unterprogramm:

```
1000 PRINT "I =" ; I , "X =" ; X
1010 GET A$ : IF A$ <>CHR$(32) THEN 1010
1020 RETURN
```

In die erste FOR-NEXT-Schleife fügen wir direkt hinter die READ-Anweisung einen Aufruf dieses Unterprogramms ein:

```
125 GOSUB 1000
```

Wenn wir das Programm nun laufenlassen, geschieht folgendes: Der Computer gelangt mit Zeile 110 in die Leseschleife. In Zeile 120 wird jeweils ein DATA-Element gelesen. Dann erfolgt mit der eingefügten Zeile 125 ein Sprung in das vorhin geschriebene Unterprogramm. Dieses Unterprogramm druckt den Wert der Zählvariablen I und den soeben gelesenen DATA-Wert X aus und wartet dann, bis die Leertaste betätigt wird. Dann kehrt das Unterprogramm zurück und die Schleife wird nach dem NEXT in Zeile 140 erneut durchlaufen.
Auf diese Art und Weise erhält man am Bildschirm eine übersichtliche Darstellung der gelesenen DATA-Werte, die man leicht mit der Vorlage vergleichen kann. Wenn wir das Programm jetzt starten, erhalten wir jeweils nach Drücken der Leertaste eine Bildschirmanzeige etwa in der folgenden Art:

```
I = 1      X = 12
I = 2      X = 33
I = 3      X = 11
```

und so weiter bis schließlich das Ende von DATA-Zeile 310 erreicht wird:

```
I = 9      X = 18
I = 10     X = 0
```

Nanu? Das hatten wir eigentlich nicht erwartet. X=18 ist der letzte Wert in Zeile 310 und danach sollte eigenlich der erste Wert aus der nächsten DATA-Zeile gelesen werden, nämlich X=11. Woher also kommt dieser Wert Null bei I=10? Ein Vergleich von Zeile 310 in Listing 1 (abgetippt) mit Listing 2 (Original) führt uns auf des Rätsels Lösung. Offenbar haben wir beim Abtippen am Ende von Zeile 310 noch ein Komma gesetzt, was da nicht hingehört. Ein Komma in einer DATA-Anweisung trennt für unseren Commodore-Computer aber immer zwei Werte voneinander, und da er hinter dem letzten Komma nichts mehr findet, setzt er kurzerhand den Wert Null dafür an.
Damit haben wir den überzähligen DATA-Wert im ersten Block gefunden. Wir entfernen das Komma in Zeile 310 und löschen die Zeile 125 mit dem GOSUB-Befehl und ebenso die Zeile 145 mit dem nun nicht mehr benötigten STOP-Befehl.
Ein erneuter Probelauf des Programms schreibt die Zeichenfolge »COMMOORE« links oben in den Bildschirm – und bringt die Fehlermeldung »OUT OF DATA ERROR IN 220«. Nach Auflisten der Zeile 220 sehen wir leider nur

```
220 READ X
```

```
1 REM DATA-TEST
2 REM ---------
3 REM
4 REM
5 B=1024:REM BILDSCHIRM
6 REM
7 PRINT"3":REM CLEAR
8 PRINT:PRINT:PRINT:PRINT
9 REM
100 REM DATA-BLOCK 1 LESEN
105 REM
110 FOR I=1 TO 17
120 READ X
130 S=S+X
140 NEXT I
150 PRINT"S =";S
160 YF S<>282 THEN PRINT"PRUEFSUMMENFEHLER":END
170 PRINT"OK"
190 REM
195 REM
200 REM DATA-BLOCK 2 LESEN
205 REM
210 FOR I=0 TO 8
220 READ X
230 POKE B+I,X
240 NEXT I
250 END
290 REM
295 REM
300 REM DATA-BLOCK 1
305 REM
310 DATA 12,33,11,4,17,38,22,19,7,18,
320 DATA 11,41,15,19,3,12,0
345 REM
350 REM DATA-BLOCK 2
355 REM
360 DATA 3,1513,13,15.4,15,18,5

READY.
```

Listing 1. In diesem Testprogramm sind einige Fehler in den DATA-Zeilen enthalten

```
290 REM
295 REM
300 REM DATA-BLOCK 1
305 REM
310 DATA 12,33,11,4,17,38,22,19,7,18
320 DATA 11,41,15,19,3,12,0
345 REM
350 REM DATA-BLOCK 2
355 REM
360 DATA 3,15,13,13,15,4,15,18,5

READY.
```

Listing 2. Dies ist nochmals der DATA-Block aus Listing 1, aber diesmal das fehlerfreie »Original«

```
100 DATA 12,13,14,15,16,17,0
200 DATA 13,12,14,15,16,17,0
300 DATA 13,12,14,15,16,17,0,0,0
400 DATA 23,2,14,15,16,17,0
500 DATA 23,2,14,15,16,,17

READY.
```

Listing 3. Die Bildung einer Prüfsumme ist kein zuverlässiges Mittel, um Fehler in DATA-Zeilen zu entdecken. Wie man leicht nachrechnen kann, ergibt jede der fünf DATA-Zeilen die gleiche Prüfsumme.

Das bringt uns nicht viel weiter. Die Fehlermeldung und das verstümmelte Wort »COMMODORE« am oberen Bildschirmrand deuten aber auf einen fehlenden DATA-Wert in Block 2 hin. Untersuchen wir also Block 2 einmal genauer. Das Unterprogramm zum Ausdrucken der Werte von I und X am Bildschirm befindet sich ja ab Zeile 1000 noch im Speicher. Wir brauchen daher nur einen entsprechenden GOSUB-Befehl in die zweite Leseschleife einzufügen, am besten gleich nach dem READ-Befehl, also etwa in Zeile 225:

```
225 GOSUB 1000
```

Wir erhalten wieder eine leicht zu überprüfende Liste aller DATA-Werte, diesmal aus Block 2. Bei I=4 fällt uns sofort etwas auf. Am Bildschirm erscheint nämlich

```
I = 4      X = 15.4
```

Das ist die einzige Zahl mit Nachkommastellen, was bei dieser Art der Bildschirmausgabe sofort ins Auge sticht. Wir vergleichen den Wert mit der Eintragung im Originallisting und sehen sofort den Fehler: Wir haben beim Abtippen irrtümlich einen Punkt statt eines Kommas eingegeben. Die Korrektur ist leicht ausgeführt.

Danach löschen wir die jetzt überflüssige Zeile 225 mit dem GOSUB 1000 wieder und überzeugen uns durch einen abschließenden Probelauf vom einwandfreien Funktionieren des Programms.

Natürlich kann man nicht erwarten, daß sich alle Fehler so reibungslos lokalisieren lassen wie in unserem kleinen Beispiel. Gerade bei Fehlern in DATA-Zeilen kann die Suche sich namentlich bei längeren DATA-Blöcken um einiges schwieriger gestalten. Aber bei Programmen mit vielen DATA-Zeilen sind die hier beschriebenen Methoden zum Auffinden von versteckten Fehlern einfach unentbehrlich, wenn man einigermaßen schnell und sicher zum Ziel gelangen will.

Das Optimum herausholen

Zum Schluß soll noch vor einer verbreiteten Unsitte gewarnt werden, die für viele fehlerhafte Programme verantwortlich ist. Gemeint ist das Optimieren von Programmen hinsichtlich Speicherbedarf, Geschwindigkeit, Benutzerfreundlichkeit oder eines anderen Kriteriums.

Vielfach wird dabei versucht, während des Abtippens eines Programmes gewisse Dinge zu ändern, eben zu optimieren. Da werden Variablennamen geändert, mehrere Zeilen zu einer einzigen zusammengefaßt oder andere PRINT-Anweisungen eingebaut.

Da man ein komplexes Programm jedoch nicht so einfach überblicken kann, entstehen durch solche Maßnahmen nur zusätzliche Fehler. Es gibt zwei bekannte Programmierer-Faustregeln zum Optimieren von Software:

1. Optimiere nicht.
2. Optimiere noch nicht.

Die erste Regel kann man auch so ausdrücken: Wenn das Programm läuft und alles das tut, was es tun soll – warum um alles in der Welt sollte man es dann optimieren und sich zusätzliche Arbeit machen?

Die zweite Regel ist eine Warnung, mit dem Optimieren zu beginnen, bevor das Programm in seiner ursprünglichen Fassung lauffähig ist.

Wenn Sie alle hier beschriebenen Regeln und Methoden bei der Suche von Fehlern wirklich beherzigen, dann sollte es Ihnen auch als Programmierneuling möglich sein, selbstständig Fehler in Ihren Programmen auszumerzen. (ev)

Relative Dateien leicht verständlich

Die Datenverwaltung zählt – neben dem Spielen – wohl zu den häufigsten Anwendungsgebieten für Heimcomputer. Um Daten aber effektiv verwalten zu können, bedarf es des sinnvollen Einsatzes bestimmter Verfahren und Techniken.

Die relative Dateiorganisation, die sich vor allem bei der Verwaltung größerer Datenbestände als sehr nützlich erweist, ist eine davon und wird im folgenden Artikel näher erläutert.

Zunächst ein paar Worte zum Begriff der »Datei«. Eine Datei läßt sich am besten vergleichen mit einem Karteikasten mit vielen gleichartigen Karteikarten. Einer Karteikarte entspricht dabei ein sogenannter »Datensatz« der Datei. Jeder Datensatz unterteilt sich wiederum in verschiedene »Datenfelder«, in die die einzelnen Eintragungen auf der Karteikarte erfolgen.

Ein kleines Beispiel: In Bild 1 links sehen Sie, wie ein Eintragungs- bzw. Eingabeschema zur Verwaltung von Adressen aussehen könnte. Die Punkte zwischen den Ausrufezeichen markieren die Stellen, an die die einzelnen Daten jeweils geschrieben werden, wie Sie in Bild 1 rechts sehen können. Die Bereiche zwischen den Ausrufezeichen stellen also die einzelnen Datenfelder dar! Alle Inhalte der Datenfelder zusammen (das heißt eine komplette Adresse bestehend aus Nachname, Vorname, Straße, Wohnort und Telefon) ergeben dann einen Datensatz; mehrere gleichartige Datensätze eine Datei. (Eine schematische Darstellung dieses Sachverhalts finden Sie in Bild 2.)

Um nun Daten mit dem Computer zu verwalten, gibt es beim Commodore 64 in Verbindung mit der Floppy VC 1541 grundsätzlich zwei Dateiarten: Die sequentielle und eben die relative.

Bei der sequentiellen Dateiorganisation werden alle Datensätze hintereinander, das heißt sequentiell gespeichert. Wenn nun Datensätze (in unserem Beispiel Adressen) gelesen, hinzugefügt, gelöscht oder geändert werden sollen – diese Vorgänge bezeichnet man übrigens als Dateiarbeit oder »Datenpflege« –, dann muß dazu die komplette Datei in den Rechner geladen und nach Beendigung der Datenpflege wieder auf die Diskette zurückgespeichert werden. Ein direkter Zugriff auf jeden einzelnen Datensatz der Datei ist nicht möglich.

Dieser Umstand ist einerseits sehr zeitraubend (die Floppy 1541 ist ja nicht gerade die Schnellste), andererseits macht er die maximale Dateigröße vom verfügbaren Speicherplatz im Rechner abhängig. Die sequentielle Datenspeicherung ist daher mehr etwas für kleinere Dateien (mit wenigen Datensätzen) oder für Dateien mit sehr kurzen Datensätzen (was wir uns später noch zunutze machen werden).

Wesentlich flexibler in bezug auf die Datenpflege ist die relative Dateiorganisation, die ja der eigentliche Gegenstand dieses Artikels ist. Bei ihr ist es nämlich möglich, direkt auf jeden einzelnen Datensatz – auch »Record« genannt – der

64er-online.de 64er-online.net

```
Nachname: !............................!
Vorname:  !....................!
Strasse:  !............................!
Wohnort:  !............................!
Telefon:  !....................!
```

Ein Eintragungs- beziehungsweise Eingabeschema zur Verwaltung von Adressen . . .

```
Nachname: !Mueller......................!
Vorname:  !Juergen.............!
Strasse:  !Waldstr. 1..................!
Wohnort:  !Frankfurt 1.................!
Telefon:  !1234/677889.........!
```

. . . und eine Beispieleintragung.

Bild 1. Eine Adressenverwaltung als Beispiel einer Datei

Der Aufbau einer Datei:
(allgemein)

Feld1	Feld2	Feld3	Feld4	Feld5	Feld1	Feld2	Feld3
Datensatz5					Datensatz6		
DATEI							

(im Beispiel)

Haus	Hans	Waldstr. 2	Astadt	123456	Mueller	Juergen	Waldstr. 1
Adresse5					Adresse6		
ADRESSEN							

Bild 2. So ist eine Datei im Prinzip aufgebaut

Datei zuzugreifen! (Zusätzlich kann sogar auf eine beliebige Stelle innerhalb eines Records positioniert werden.)

Zum Prinzip der relativen Datei

Bei der relativen Dateiorganisation werden alle Datensätze (Records) in laufender Reihenfolge durchnumeriert. Der erste Datensatz bekommt dabei die Nummer 1, der zweite die Nummer 2, etc. Wenn Sie jetzt auf einen Datensatz zugreifen wollen, müssen Sie nur noch dessen Nummer angeben.

Dieser Vorteil ist allerdings mit einer wichtigen Bedingung verbunden:

Alle Records einer Datei müssen eine konstante, beim Einrichten der Datei festzulegende Länge zwischen 1 und 254 Byte (Zeichen) aufweisen!

Diese Bedingung wird verständlich, wenn man sich überlegt, wie das Floppy-Betriebssystem (kurz: Floppy-DOS) aus der Recordnummer die Position des Datensatzes auf der Diskette bestimmt.

Angenommen, Sie wollen auf den 27. Record einer Datei mit einer Recordlänge von 50 Byte zugreifen. Um nun die Position dieses Datensatzes zu ermitteln, multipliziert das Floppy-DOS die Recordlänge (in unserem Fall also 50) mit der Recordnummer -1 (also 26), was die Anzahl der Byte ergibt, die von den ersten 26 Records der Datei belegt werden (=1300 Byte). Dieser Wert wird jetzt noch zur Anfangsposition der Datei hinzuaddiert, was schließlich die Position des gewünschten Datensatzes ergibt. (Die ersten 26 Records werden durch das Aufaddieren ihrer Gesamtlänge auf den Anfang der Datei praktisch »übersprungen«.)

Jetzt wird auch der Name dieser Dateiart, nämlich »relativ«, verständlich, denn die Record-Position wird ja relativ zum Anfang der Datei ermittelt! (Im Beispiel beginnt der gewünschte Datensatz 1300 Byte vom Anfang der Datei entfernt.)

Wie gesagt kann also jeder Datensatz einer relativen Datei durch Angabe seiner Nummer direkt angesprochen werden. Wenn Sie aber zum Beispiel eine bestimmte Adresse suchen, werden Sie in der Regel kaum die Nummer des betreffenden Datensatzes kennen. Das gleiche gilt auch für die meisten anderen Anwendungsgebiete. (Höchstens bei einer Kontenverwaltung – zum Beispiel für ein Haushaltsbuch – wäre eine direkte Verwendung der Recordnummer sinnvoll.) Daher verwendet man in der Praxis häufig eine Mischform aus sequentieller und relativer Dateiorganisation, die »indexsequentielle Datei«.

Das Prinzip ist ganz einfach:

Alle Daten werden – wie bisher – in einer relativen Hauptdatei angelegt. Zusätzlich wird dann eine sequentielle Index- oder Schlüsseldatei angelegt. Diese Schlüsseldatei enthält von jedem Datensatz der relativen Datei nur einen Teil – bei einer Adreßdatei zum Beispiel den Nachnamen – sowie die Recordnummer des zugehörigen Datensatzes der relativen Hauptdatei, in dem – im Beispiel – die komplette Adresse untergebracht ist.

Wenn Sie jetzt beispielsweise nach der Adresse eines Herrn Müller suchen, wird einfach die Schlüsseldatei komplett in den Computer geladen (da sie sehr kurz ist, geht dies ja relativ schnell, und auch die Maximalanzahl der Adressen ist verhältnismäßig hoch), dort durchsucht und – sofern der Datensatz »Müller« vorhanden ist – auf den bei »Müller« (in Form der Recordnummer) vermerkten Datensatz positioniert. Dieser wird dann schließlich in den Speicher geladen und ausgegeben.

Natürlich ist es auch möglich, zu einer relativen Hauptdatei mehrere Indexdateien anzulegen (zum Beispiel eine mit den Nachnamen als Index, eine mit den Vornamen, etc.). Allerdings muß man dann das eventuell wieder auftretende Speicherplatzproblem – die Indexdateien müssen ja komplett in den Speicher geladen werden – im Auge behalten beziehungsweise muß sich vor der Dateiarbeit entscheiden, mit welcher Indexdatei man jeweils arbeiten will.

Nach all der Theorie kommen wir nun zur Praxis:

Wie werden relative Dateien mit dem Commodore 64 und der Floppy VC 1541 – die Anlage relativer Dateien auf Kassette ist nicht möglich! – angelegt und verwaltet? Mit dieser Frage wollen wir uns im Folgenden ausführlich befassen. Zum Abschluß wird dann noch eine kleine, ausführlich kommentierte indexsequentielle Adreßverwaltung (Listing 2) vorgestellt, die Sie selbst nach Belieben abändern können, um so Ihre neuen Kenntnisse in der Praxis zu erproben.

Die Verwaltung einer relativen Datei gliedert sich grob in vier Abschnitte:

- »Öffnen« der Datei

- Positionieren auf den gewünschten Record
- Lesen/Schreiben des Records
- »Schließen« der Datei.

Da das Basic 2.0 des Commodore 64 leider keine speziellen Befehle zur Unterstützung von relativen Dateien zur Verfügung stellt, bleibt nur der Weg über »OPEN«- und »CHR$(...)«-Sequenzen. Daher sieht das Ganze auf den ersten (und wohl auch zweiten) Blick etwas kompliziert aus. Wenn man sich die erforderlichen Programmteile aber – wie in Listing 1 geschehen – als Unterprogramme schreibt, ist es halb so schlimm.

Kommen wir zum ersten Schritt, dem »Öffnen« der Datei.

Öffnen einer relativen Datei

Bei einer relativen Datei ist es – im Gegensatz zum Beispiel zur sequentiellen Datei – praktisch egal, ob die Datei auf der Diskette schon eingerichtet ist oder nicht oder ob Sie Daten lesen oder schreiben wollen. Der Öffnungsbefehl ist immer derselbe:

```
OPEN LF,GA,SA,DN$+",L," +CHR$(RL)
```

Die einzelnen Variablen – auch »Parameter« genannt – haben folgende Bedeutung:

LF: Das ist die logische Filenummer der Datei. Sie ist erforderlich, damit der Computer bei späteren Lese- oder Schreibbefehlen (bei denen diese Nummer angegeben werden muß) weiß, aus welcher Datei gelesen oder in welche Datei geschrieben werden soll. LF kann Werte zwischen 1 und 127 annehmen.

GA: Das ist die Geräteadresse der Floppy. Im Normalfall also 8.

SA: Die »Sekundäradresse« oder »Kanalnummer« SA hat auf Floppy-Ebene eine ähnliche Bedeutung wie die logische Filenummer LF auf Computer-Ebene.

Für uns ist aber nur wichtig, daß SA Werte zwischen 2 und 14 annehmen kann.

Wenn Sie mehrere Dateien gleichzeitig geöffnet haben, müssen Sie außerdem darauf achten, daß Sie jeder Datei eine andere Sekundäradresse zuweisen. Wird nämlich eine zweite Datei mit derselben Sekundäradresse wie die erste geöffnet, so wird die zuerst geöffnete Datei auf Floppy-Ebene geschlossen!

In diesem Zusammenhang noch ein wichtiger Hinweis:

Auf Floppy-Ebene können maximal drei Dateien gleichzeitig geöffnet sein. (Auf Computer-Ebene sind es übrigens maximal 10). Da zur Verwaltung einer relativen Datei neben der eigentlichen Datei – wie wir gleich noch sehen werden – auch der Floppy-Kommandokanal geöffnet werden muß, bedeutet das, daß maximal eine relative und eine sequentielle Datei gleichzeitig geöffnet sein können (was gerade für eine indexsequentielle Dateiorganisation reicht).

DN$: DN$ enthält den Namen der Datei. Dieser darf maximal 16 Zeichen lang sein. Außerdem ist darauf zu achten, daß kein File (=Datei oder Programm) desselben Namens bereits auf der Diskette existiert – außer natürlich eine früher definierte relative Datei dieses Namens, mit der Sie jetzt arbeiten wollen, denn sonst gibt es einen »FILE EXISTS ERROR«.

",L,": Dieser Parameter teilt dem Floppy-DOS mit, daß eine relative Datei geöffnet werden soll. (Die Kommas dienen zur Abtennung von den übrigen Parametern.)

RL: RL enthält die gewünschte Recordlänge der Datei und kann Werte zwischen 1 und 254 annehmen.

Wichtig: Die beim ersten Öffnen der Datei angegebene Recordlänge kann später nicht mehr geändert werden! (Wenn Sie versuchen, eine bereits existierende relative Datei mit einer anderen Recordlänge als bei der ersten Öffnung angegeben zu öffnen, so wird ein »RECORD NOT PRESENT ERROR« angezeigt.)

Zur Ermittlung der Recordlänge zählen Sie einfach alle Datenfeldlängen eines Datensatzes zusammen. Im Eingangsbeispiel ergibt das eine Länge von 15 (Nachname) + 10 (Vorname) + 15 (Straße) + 15 (Wohnort) + 11 (Telefon) = 66 Byte.

Sofern Sie zum Einlesen der Records in den Computer den INPUT#-Befehl benutzen, müssen Sie dazu noch ein Byte addieren, denn, um das Ende eines Records erkennen zu können, benötigt der INPUT#-Befehl ein »RETURN« (entspricht CHR$(13)) oder ein Komma (entspricht CHR$(44)) am Ende des Records. Das »RETURN« wird normalerweise am Ende eines jeden PRINT- beziehungsweise PRINT#-Befehls gesendet – sofern Sie es nicht durch ein Semikolon am Ende des PRINT-Kommandos (zum Beispiel bei »`PRINT#1, " text";`«) unterdrücken.

Im Beispiel ergibt sich also eine Gesamtlänge von 67 Byte. Nachdem die Datei geöffnet wurde, kann mit der eigentlichen Dateiarbeit begonnen werden. Um jetzt einen bestimmten Record zu schreiben beziehungsweise zu lesen, muß zunächst auf ihn positioniert werden.

Positionieren auf einen Record

Da die Positionieranweisung über den Floppy-Kommandokanal gesendet werden muß, muß dieser zuerst mit »OPEN 15,8,15,« geöffnet werden. Die erste 15 ist wieder die logische Filenummer (Um Kollisionen mit Dateien zu vermeiden, ist es zweckmäßig, hier immer dieselbe logische Filenummer zu verwenden); bei der 8 handelt es sich um die Geräteadresse der Floppy. Die zweite 15 ist die Kanalnummer des Floppy-Kommandokanals.

Der Positionierbefehl ist ähnlich kompliziert aufgebaut wie der OPEN-Befehl:

```
PRINT#15,"P"+CHR$(SA)+CHR$(LB)+CHR$(HB)+CHR$(RP)
```

Die Parameter im einzelnen:

SA: Das ist die beim Öffnen der Datei festgelegte Sekundäradresse.

LB/HB: Das ist die Recordnummer in einer für Sie vielleicht etwas ungewohnten Form, der sogenannten Lowbyte-/Highbyte-Form.

Da mit CHR$(...) nur Werte bis zu 255 übermittelt werden können, muß die Recordnummer in zwei solcher Werte aufgeteilt werden, die sich aus folgenden Formeln ergeben: HB=INT(RN/256):LB=RN-256*HB

RP: Mit RP können Sie auf eine bestimmte Stelle innerhalb des Records positionieren. Wollen Sie zum Beispiel einen Record ab dem 10. Zeichen einlesen, so muß RP den Wert 10 erhalten.

Wichtig: Eine Positionierung auf eine Stelle innerhalb eines Records ist nur beim Lesen dieses Records sinnvoll. Zum Schreiben eines Records muß RP auf 1 gesetzt werden!

Und natürlich ist auch nur eine solche Positionsangabe sinnvoll, die kleiner gleich der Recordlänge ist. Daher ergibt sich für RP – eine entsprechende Recordlänge vorausgesetzt – ein Maximalwert von 254.

Nach der Positionierung ist der betreffende Record nun für einen Lese- oder Schreibzugriff bereit.

Schreiben eines Records

Vor dem Schreiben eines Records sind drei Dinge zu beachten:

- Es muß unbedingt auf das erste Byte des Records positioniert werden (RP=1).
- Es sollte sichergestellt werden, daß alle Datenfelder – und

damit auch der Record – ihre vorgeschriebene Länge aufweisen. Wie eine solche Kontrolle und Korrektur aussehen könnte, sehen Sie in Listing 2 in den Zeilen 860–920.
- Da der Record in einem Stück geschrieben werden muß, müssen die einzelnen Datenfelder vor dem Schreiben zusammengefaßt werden (siehe auch Listing 2 Zeile 940 bis 960).

Sind diese Voraussetzungen erfüllt, so kann der Record mit folgendem Befehl geschrieben werden:

```
PRINT#LF,RC$
```

LF ist dabei die beim OPEN-Befehl festgelegte logische Filenummer; RC$ enthält den zu schreibenden Record.

In diesem Zusammenhang erwähnenswert ist die Fehlermeldung »RECORD NOT PRESENT ERROR«, was soviel bedeutet wie »Datensatz nicht vorhanden«. Falls nämlich auf einen Record zugegriffen (geschrieben/gelesen) wird, der zuvor noch nicht beschrieben wurde, so wird diese Meldung erzeugt. Beim (erstmaligen) Schreiben eines Records kann diese Meldung natürlich ignoriert werden. (Der Schreibbefehl wird trotz der Meldung ausgeführt.)

Weiterhin ist zu beachten: Ein Schreibbefehl auf einen zuvor noch nicht beschriebenen Record hat zur Folge, daß alle davor liegenden (zuvor noch nicht beschriebenen) Records ebenfalls beschrieben werden!

Ein Beispiel: Sie haben eine relative Datei bereits bis zum 20. Record beschrieben. Nun schreiben Sie den 100. Record. Dadurch werden gleichzeitig die Records 21 bis 99 (mit demselben Inhalt wie der 100. Record) beschrieben!

Wenn Sie also eine relative Datei nicht der Reihe nach beschreiben wollen, dann sollten Sie beim Einrichten der Datei als erstes den Record mit der höchsten Nummer (mit einem beliebigen Inhalt) beschreiben. Dadurch bleiben Ihnen unnötige Wartezeiten bei der Dateiarbeit – das Beschreiben von zum Beispiel 50 Records nimmt einige Zeit in Anspruch – erspart.

Lesen eines Records

Zum Lesen eines Records gibt es zwei Möglichkeiten:

Die erste Möglichkeit stellt der INPUT#-Befehl dar. Mit ihm kann ein Record komplett in den Computer geladen werden – unter zwei Bedingungen: Der Record darf nicht länger als 88 Zeichen sein, und er muß ein »RETURN« oder ein Komma als letztes Zeichen aufweisen.

Das Befehlsformat: `INPUT#LF,RC$`

LF ist dabei die beim OPEN-Befehl festgelegte logische Filenummer; RC$ wird der Record zugewiesen.

Eine zweite Möglichkeit – bei einer Recordlänge von mehr als 88 Byte zudem die einzige – bietet der GET#-Befehl. Um zum Beispiel einen 100 Byte langen Record einzulesen, wäre folgende Befehlssequenz denkbar:

```
FOR I=1 TO 100:GET#LF,EG$:RC$=RC$+EG$: NEXT I
```

LF ist dabei wieder die beim OPEN-Befehl festgelegte logische Filenummer.

Schließen einer relativen Datei

Nach Beendigung der Dateiarbeit muß die Datei geschlossen werden. Der dazu erforderliche Befehl ist – ausnahmsweise – sehr einfach.

```
CLOSE LF
```

LF ist dabei wiederum die beim OPEN-Befehl festgelegte logische Filenummer.

Zum Abschluß eine kleine Adreßverwaltung zum Experimentieren. Bitte geben Sie dazu Listing 2 ein.

Wichtig: Das Programm ist nur mit den in Listing 1 enthaltenen Unter-Programmen lauffähig. Tippen Sie also am besten zuerst Listing 1 ein und speichern es (für spätere eigene Entwicklungen, für die Sie die Unterprogramme gut gebrauchen können). Danach geben Sie zusätzlich Listing 2 ein und speichern das Gesamtprogramm dann ebenfalls auf Diskette.

Da das Programm selbst ausführlich dokumentiert ist, möchte ich an dieser Stelle nur noch kurz ein paar Bedienungshinweise geben:

Nachdem das Hauptmenü ausgegeben wurde, geben Sie als erstes »1« (+"RETURN") ein. Dadurch werden alle notwendigen Vorbereitungen für die Dateiarbeit getroffen. Sobald das Hauptmenü wieder erschienen ist, können Sie wahlweise (und auch abwechselnd) Adressen ein- oder ausgeben. Bei der Ausgabe wird dabei der Nachname als Index verwendet. Nach Beendigung der Dateiarbeit geben Sie »4« ein, wodurch die Datei ordnungsgemäß geschlossen und das Programm beendet wird.

Das Programm ist zwar recht spartanisch, dafür aber sehr flexibel. Beides soll Sie dazu animieren, selbst tätig zu werden und das Programm nach Ihren speziellen Wünschen abzuändern und zu ergänzen. Denn auch bei der Datenverwaltung gilt, wie überall: Übung macht den Meister!

(Martin Hecht/gk)

```
1680 REM--- UNTERPROGRAMME ZUR VERWALTUNG
     VON RELATIVEN DATEIEN ---------------
     -                                      <016>
1740 :                                      <192>
1750 REM  LF = LOGISCHE FILENUMMER          <152>
1760 REM  SA = SEKUNDAERADRESSE             <169>
1770 REM  DN$= DATEINAME                    <120>
1780 REM  RL = RECORDLAENGE                 <242>
1790 :                                      <242>
1800 OPEN 15,8,15:REM FLOPPY-KOMMANDOKANAL
      OEFFNEN                               <163>
1810 OPEN LF,8,SA,DN$+",L,"+CHR$(RL):REM D
     ATEI OEFFNEN                           <041>
1820 :                                      <018>
1830 RETURN                                 <110>
1840 :                                      <038>
1850 :                                      <048>
1860 :                                      <058>
1870 REM--- POSITIONIEREN AUF EINEN RECORD
     -----------------------------------
     -                                      <020>
1880 :                                      <078>
1890 REM  SA = SEKUNDAERADRESSE             <045>
1900 REM  RN = RECORDNUMMER                 <199>
1910 REM  RP = POSITION INNERHALB DES RECO
     RDS                                    <167>
1920 :                                      <118>
1930 HB=INT(RN/256):LB=RN-256*HB:REM RECOR
     DNUMMER IN LOW/HIGH AUFTEILEN          <060>
1940 PRINT#15,"P"+CHR$(SA)+CHR$(LB)+CHR$(H
     B)+CHR$(RP)                            <089>
1950 :                                      <148>
1960 RETURN                                 <240>
1970 :                                      <168>
1980 :                                      <178>
1990 :                                      <188>
2000 REM--- SCHLIESSEN EINER (RELATIVEN) D
     ATEI --------------------------------
     -                                      <088>
2010 :                                      <208>
2020 REM  LF = LOGISCHE FILENUMMER          <168>
2030 :                                      <228>
2040 CLOSE LF:REM RELATIVE DATEI SCHLIESSE
     N                                      <234>
2050 CLOSE 15:REM FLOPPY-FEHLERKANAL SCHLI
     ESSEN                                  <171>
2060 :                                      <004>
2070 RETURN                                 <096>
```

© 64'er

Listing 1. Unterprogramme zur Verwaltung einer relativen Datei

```
100 REM--- ADRESSVERWALTUNG MIT INDEXSEQUENTIELLER D
    ATEIORGANISATION -----------                    <217>
110 REM--- (W) 1985 BY MARTIN HECHT,STGT -----------
    ---------------------------                     <132>
120 :                                               <096>
130 :                                               <106>
140 :                                               <116>
150 REM--- DIMENSIONIERUNGEN ----------------------
    ---------------------------                     <043>
160 :                                               <136>
170 DM=100:REM MAXIMALE DATENSATZANZAHL             <056>
180 :                                               <156>
190 DIM ID$(DM):REM INDEXFELD (ENTHAELT NACHNAMEN)  <146>
200 DIM IN(DM):REM ENTHAELT ZUGEHOERIGE RECORDNUMMER
    N                                               <086>
210 DIM DS$(5):REM DATENFELDER                      <063>
220 :                                               <196>
230 :                                               <206>
240 :                                               <216>
250 REM--- HAUPTMENUE ----------------------------
    ---------------------------                     <249>
260 :                                               <238>
270 WL=0:PRINT:PRINT                                <199>
280 PRINT TAB(11) "ADRESSENVERWALTUNG"              <039>
290 PRINT                                           <138>
300 PRINT TAB(8)"-1-: DATEIARBEIT BEGINNEN"         <048>
310 PRINT TAB(8)"-2-: ADRESSEN EINGEBEN"            <173>
320 PRINT TAB(8)"-3-: ADRESSEN AUSGEBEN"            <250>
330 PRINT TAB(8)"-4-: DATEIARBEIT BEENDEN"          <108>
340 PRINT                                           <188>
350 :                                               <072>
360 PRINT TAB(11) "IHRE WAHL (1-4)";:INPUT WL       <166>
370 :                                               <092>
380 IF WL=1 THEN GOSUB 490:REM DATEIARBEIT BEGINNEN <180>
390 IF WL=2 THEN GOSUB 710:REM ADRESSEN EINGEBEN    <036>
400 IF WL=3 THEN GOSUB 1110:REM ADRESSEN AUSGEBEN   <132>
410 IF WL=4 THEN GOTO 1480:REM DATEIARBEIT BEENDEN  <099>
420 :                                               <142>
430 GOTO 270:REM ZURUECK ZUM HAUPTMENUE             <123>
440 :                                               <162>
450 :                                               <172>
460 :                                               <182>
470 REM--- DATEIARBEIT BEGINNEN ------------------
    ---------------------------                     <005>
480 :                                               <202>
490 LF=1:SA=2:DN$="RELADR":RL=67:GOSUB 1800:REM RELA
    TIVE DATEI 'RELADR' OEFFNEN                     <134>
500 :                                               <222>
510 OPEN 2,8,3,"SEQADR,S,R":REM SEQUENTIELLE DATEI
    SEQADR' ZUM LESEN OEFFNEN                       <156>
520 :                                               <244>
530 GOSUB 1620:REM FLOPPY-FEHLERKANAL AUSLESEN      <096>
540 IF ER<>0 THEN AD=0:GOTO 590:REM DATEI IST NOCH N
    ICHT ANGELEGT                                   <154>
550 :                                               <018>
560 INPUT#2,AD:REM ANZAHL DER DATENSAETZE           <204>
570 FOR I=1 TO AD:INPUT#2,ID$(I),IN(I):NEXT I:REM IN
    DEXDATEI EINLESEN                               <101>
580 :                                               <048>
590 FL=1:REM KENNZEICHNUNG FUER 'DATEI IM RECHNER'  <189>
600 :                                               <068>
610 CLOSE 2:REM SEQUENTIELLE DATEI SCHLIESSEN       <147>
620 :                                               <088>
630 PRINT:PRINT:PRINT TAB(7) "DATEIARBEIT KANN BEGIN
    NEN !"                                          <032>
640 :                                               <108>
650 RETURN                                          <200>
660 :                                               <128>
670 :                                               <138>
680 :                                               <148>
690 REM--- ADRESSEN EINGEBEN --------------------
    ---------------------------                     <049>
700 :                                               <168>
710 IF FL=0 THEN PRINT:PRINT TAB(8) "KEINE DATEI IM
    RECHNER !":RETURN                               <110>
720 :                                               <188>
730 REM ADRESSE EINGEBEN                            <009>
740 PRINT:PRINT                                     <184>
750 INPUT "NACHNAME (15) :";DS$(1)                  <225>
760 INPUT "VORNAME (10) :";DS$(2)                   <135>
770 INPUT "STRASSE (15) :";DS$(3)                   <123>
780 INPUT "WOHNORT (15) :";DS$(4)                   <160>
790 INPUT "TELEFON (11) :";DS$(5)                   <078>
800 :                                               <119>
810 REM ADRESSE IN INDEXDATEI VERMERKEN             <065>
820 AD=AD+1:REM ANZAHL DER ADRESSEN UM 1 ERHOEHEN   <107>
830 ID$(AD)=DS$(1):REM NACHNAME                     <135>
840 IN(AD)=RN:REM RECORDNUMMER                      <131>
850 :                                               <064>
860 REM ADRESSFELDER AUF RICHTIGE LAENGE BRINGEN    <218>
870 LE$="{15SPACE}":REM LEERSTRING ZUM AUFFUELLEN DE
    R FELDER                                        <190>
880 DS$(1)=LEFT$(DS$(1)+LEFT$(LE$,ABS(15-LEN(DS$(1))
    )),15)                                          <234>
890 DS$(2)=LEFT$(DS$(2)+LEFT$(LE$,ABS(10-LEN(DS$(2))
    )),10)                                          <129>
900 DS$(3)=LEFT$(DS$(3)+LEFT$(LE$,ABS(15-LEN(DS$(3))
    )),15)                                          <143>
910 DS$(4)=LEFT$(DS$(4)+LEFT$(LE$,ABS(15-LEN(DS$(4))
    )),15)                                          <099>
920 DS$(5)=LEFT$(DS$(5)+LEFT$(LE$,ABS(11-LEN(DS$(5))
    )),11)                                          <005>
930 :                                               <144>
940 REM ADRESSE FUER SPEICHERUNG ZUSAMMENFASSEN     <129>
950 RC$=""                                          <080>
960 FOR I=1 TO 5:RC$=RC$+DS$(I):NEXT I              <083>
970 :                                               <184>
980 REM ADRESSE SPEICHERN                           <233>
990 RN=AD:RP=1:GOSUB 1930:REM AUF RECORD (NR.=AD) PO
    SITIONIEREN                                     <142>
1000 PRINT#LF,RC$:REM ADRESSE IN RECORD SCHREIBEN   <249>
1010 GOSUB 1600:REM FLOPPY-FEHLERKANAL AUSLESEN     <002>
1020 :                                              <234>
1030 PRINT:PRINT:PRINT TAB(5)"ADRESSE IST GESPEICHER
     T !"                                           <040>
1040 :                                              <000>
1050 RETURN                                         <092>
1060 :                                              <020>
1070 :                                              <030>
1080 :                                              <040>
1090 REM--- ADRESSEN AUSGEBEN --------------------
     ---------------------------                    <207>
1100 :                                              <060>
1110 IF FL=0 THEN PRINT:PRINT TAB(8) "KEINE DATEI IM
      RECHNER !":RETURN                             <002>
1120 :                                              <080>
1130 PRINT:PRINT:INPUT "NACHNAME";NN$:REM INDEX ERFR
     AGEN                                           <201>
1140 :                                              <100>
1150 FOR I=1 TO AD:REM INDEXDATEI DURCHSUCHEN       <115>
1160 IF NN$=ID$(I) THEN GN=I:I=AD:NEXT I:GOTO 1220:R
     EM INDEX GEFUNDEN                              <196>
1170 NEXT I                                         <238>
1180 :                                              <140>
1190 PRINT:PRINT:PRINT TAB(5) "ADRESSE IST NICHT VOR
     HANDEN !"                                      <212>
1200 RETURN                                         <242>
1210 :                                              <170>
1220 RN=GN:RP=1:GOSUB 1930:REM AUF RECORD (NR. = IN(
     I)) POSITIONIEREN                              <184>
1230 INPUT#LF,RC$:REM RECORD EINLESEN               <029>
1240 :                                              <200>
1250 REM RECORD AUFTEILEN                           <005>
1260 DS$(1)=MID$(RC$,1,15):REM NACHNAME             <158>
1270 DS$(2)=MID$(RC$,16,10):REM VORNAME             <080>
1280 DS$(3)=MID$(RC$,26,15):REM STRASSE             <133>
1290 DS$(4)=MID$(RC$,41,15):REM WOHNORT             <155>
1300 DS$(5)=MID$(RC$,56,11):REM WOHNORT             <074>
1310 :                                              <016>
1320 REM RECORD AUSGEBEN                            <011>
1330 PRINT:PRINT                                    <012>
1340 PRINT "NACHNAME: ";DS$(1)                      <025>
1350 PRINT "VORNAME: ";DS$(2)                       <089>
1360 PRINT "STRASSE: ";DS$(3)                       <111>
1370 PRINT "WOHNORT: ";DS$(4)                       <087>
1380 PRINT "TELEFON: ";DS$(5)                       <073>
1390 :                                              <096>
1400 PRINT:PRINT:PRINT "WEITER MIT 'SPACE' !"       <131>
1410 GET T$:IF T$<>CHR$(32) THEN 1410:REM WARTEN AUF
      'RETURN'                                      <069>
1420 :                                              <126>
1430 RETURN                                         <218>
1440 :                                              <146>
1450 :                                              <156>
1460 :                                              <166>
1470 REM--- DATEIARBEIT BEENDEN ------------------
     ---------------------------                    <022>
1480 PRINT#15,"S:SEQADR":REM INDEXDATEI AUF DISKETTE
      LOESCHEN                                      <222>
1490 :                                              <196>
1500 OPEN 2,8,3,"SEQADR,S,W":REM SEQ. DATEI 'SEQADR'
      ZUM SCHREIBEN OEFFNEN                         <111>
1510 PRINT#2,AD:REM ANZAHL DER DATENSAETZE          <216>
1520 FOR I=1 TO AD:PRINT#2,ID$(I)","IN(I):NEXT I:REM
      INDEXDATEI SPEICHERN                          <192>
1530 :                                              <236>
1540 GOSUB 2040:REM RELATIVE DATEI SCHLIESSEN       <250>
1550 :                                              <002>
1560 END:REM PROGRAMM BEENDEN                       <250>
1570 :                                              <022>
1580 :                                              <032>
1590 :                                              <042>
1600 REM--- FLOPPY-FEHLERKANAL AUSLESEN -----------
     ---------------------------                    <223>
1610 :                                              <062>
1620 INPUT#15,ER,ER$,TR,SK                          <076>
1630 :                                              <082>
1640 RETURN                                         <174>
1650 :                                              <102>
1660 :                                              <112>
1670 :                                              <122>
```

© 64'er

Listing 2. Eine einfache Adreßverwaltung als Beispiel für eine indexsequentielle Datei

64er-online.de

Alles, was Sie schon immer über Sprites wissen wollten

Dieser Kurs führt Sie von den einfachsten Anfängen bis zu den tieferen Geheimnissen der Sprite-Programmierung. Viele Beispiele und Übungsaufgaben mit Lösungen garantieren den Lernerfolg.

In diesem Kurs wird eine der interessantesten Fähigkeiten des Commodore 64 vorgestellt: die Darstellung von Sprites. Sie werden lernen, wie ein Sprite entworfen und auf dem Bildschirm bewegt wird und auf welche Weise man seine Größe und Farbe ändern kann.

Wenn Sie ein normales Fernsehbild aus geringer Entfernung genau betrachten, werden Sie erkennen, daß es aus einer großen Anzahl winziger Punkte zusammengesetzt ist.

Auch das vom Computer erzeugte Bild besteht aus solchen Einzelpunkten. Man nennt sie sinngemäß »Bildpunkte«. Ein Bildpunkt ist das kleinste Bildelement, das der Computer auf dem Bildschirm darzustellen vermag.

Kapitel 1
Was ist ein Sprite?

Ein Sprite ist nun eine bestimmte Konfiguration von Bildpunkten, die der C 64 über den Bildschirm bewegen kann. Es besteht in seiner Grundform aus 24 Zeilen zu je 21 Punkten, insgesamt also aus 504 Bildpunkten (Bild 1).

Für jeden Bildpunkt dieser Matrix kann definiert werden, ob er sichtbar sein soll oder nicht. Damit lassen sich insgesamt

2.207.107.920.000.000.000.000.000.000.000.000.000.000.000.000.000
.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000
.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000
.000.000.000

unterschiedliche Sprites darstellen. Ihrer Phantasie sind also keine Grenzen gesetzt. Eine dieser Möglichkeiten ist in Bild 2 dargestellt.

Um ein Sprite zu definieren, müssen Sie dem Computer mitteilen, welche Bildpunkte dargestellt werden und welche unsichtbar bleiben sollen. Dazu bildet man Gruppen von jeweils acht Bildpunkten und ordnet diesen Gruppen bestimmte Zahlencodes zu.

Betrachten Sie dazu Bild 3. Jedes der acht Kästchen stellt einen Bildpunkt dar und ist durch eine Nummer gekennzeichnet. Diese Nummer ist der Wert des Bildpunktes. So hat etwa der linke Bildpunkt den Wert 128 und der rechte Bildpunkt den Wert 1.

Betrachten Sie nun Bild 4. Einige der Kästchen sind ausgefüllt. Wenn Sie die Zahlen über den ausgefüllten Kästchen addieren, erhalten Sie die Summe 85: 64 + 16 + 4 + 1 = 85. Bild 5 enthält weitere Beispiele zur Berechnung von Zahlencodes für derartige Gruppen von Bildschirmpunkten.

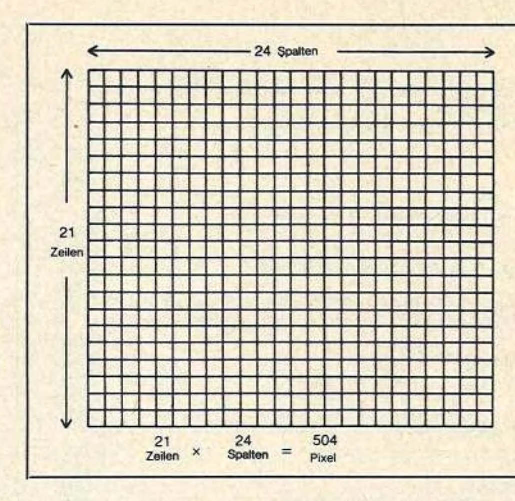


Bild 1. Ein Sprite besteht aus einer Matrix mit 504 Bildpunkten

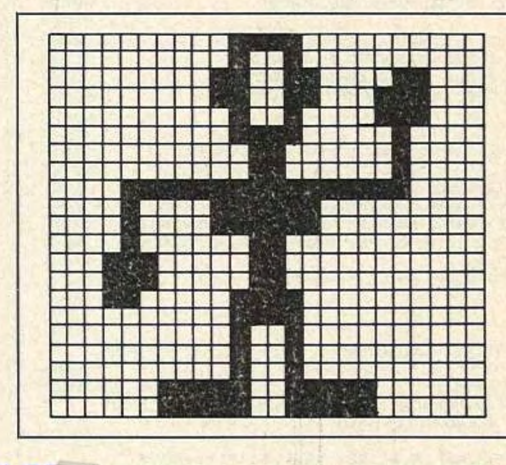


Bild 2. Ein Bild entsteht durch Sichtbarmachung einzelner Bildpunkte des Sprites



Bild 3. Bitcodes zur Codierung einer Gruppe von acht Bildpunkten

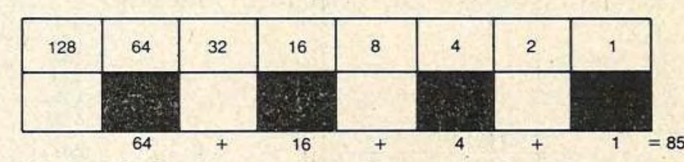


Bild 4. Die Codierung eines Musters in einer Gruppe von acht Bildpunkten

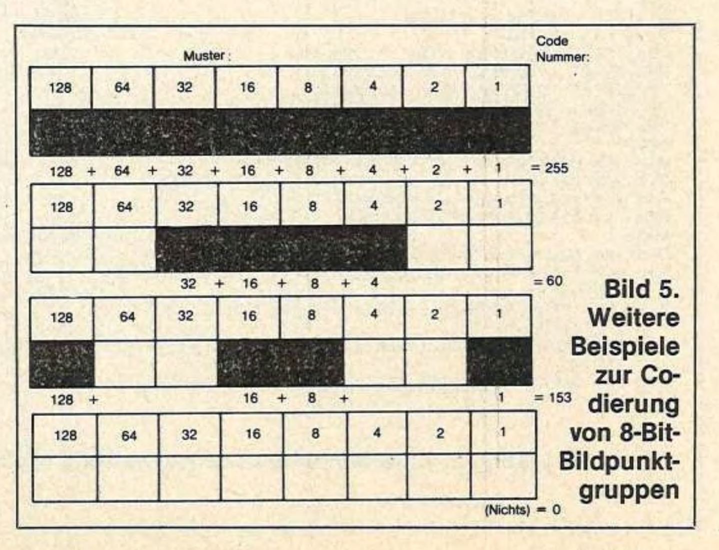


Bild 5. Weitere Beispiele zur Codierung von 8-Bit-Bildpunktgruppen

64er online
www.64er-online.de

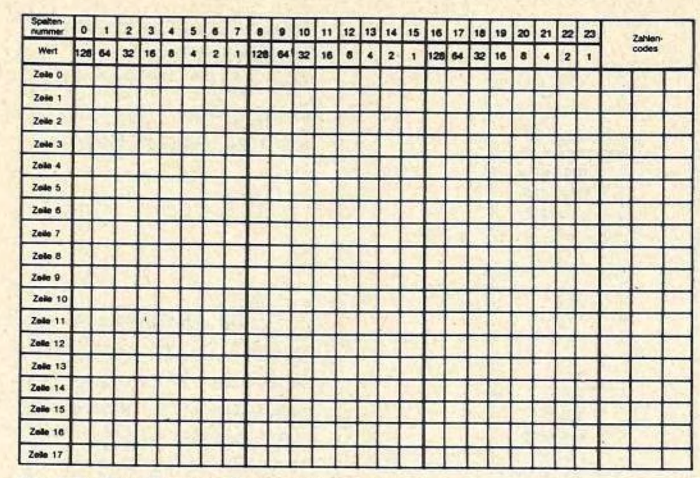


Bild 6. Ein Sprite-Entwurfsblatt

Betrachten Sie nun das Sprite-Entwurfsblatt in Bild 6. Es enthält die erforderlichen 21 Zeilen zu je 24 Bildpunkten, wobei jede Zeile in drei Gruppen zu acht Bildpunkten eingeteilt und jeder Bildpunkt mit seinem Wert versehen ist. Zusätzlich enthält jede Zeile drei Felder, in welche die Zahlencodes für die drei Gruppen zu jeweils acht Bildpunkten eingetragen werden können.

Da insgesamt 21 Zeilen zu je drei Zahlencodes vorhanden sind, wird ein Sprite also durch 63 Zahlen definiert, die Zeile für Zeile von oben nach unten und von links nach rechts in den C 64 eingegeben werden müssen. Um nun ein Sprite zu definieren, gehen Sie folgendermaßen vor:

1. Kopieren Sie sich das Sprite-Entwurfsblatt (Bild 6).
2. Entwerfen Sie ein Sprite und zeichnen Sie es ein, indem Sie für jeden darzustellenden Bildpunkt das entsprechende Kästchen ausfüllen.
3. Berechnen Sie für jede Zeile die Zahlencodes für die drei Bildpunktgruppen und tragen Sie sie ein.
4. Geben Sie die Zahlencodes in der richtigen Reihenfolge in den Computer ein.

Bild 7 stellt ein ausgefülltes Entwurfsblatt für ein kleines Sprite-Männchen dar. Lesen Sie erst weiter, wenn Ihnen der Aufbau dieses Entwurfsblattes vollständig klar ist und Sie die Berechnung der eingetragenen Werte nachvollziehen können.

Eilen Sie nun zum nächsten Kopierautomaten und machen Sie sich Kopien von dem Sprite-Entwurfsblatt. Entwerfen Sie anschließend einige Sprites. Sind Sie mit einem Entwurf zufrieden, so berechnen Sie bitte die entsprechenden 63 Zahlencodes. Sie benötigen die Daten im zweiten Kapitel.

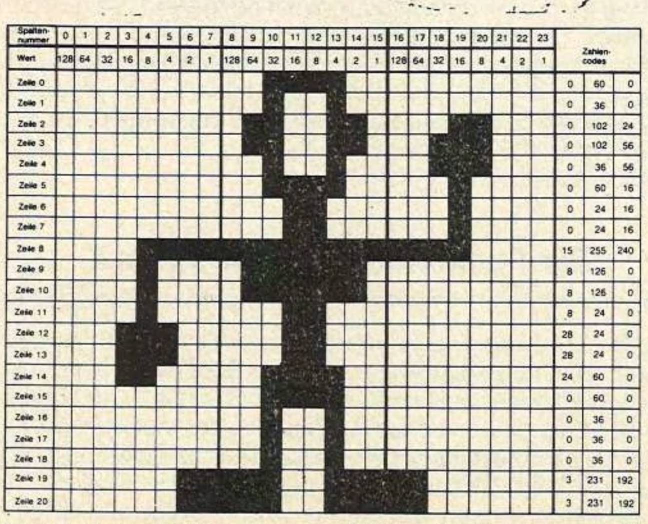


Bild 7. Beispiel für ein ausgefülltes Sprite-Entwurfsblatt

Ihr erstes Sprite-Programm

Beginnen Sie mit einem kleinen Programm zur Darstellung eines einfachen Sprites. Bild 8 zeigt den Programmaufbau, Listing 1 das Programm »Ein einfaches Sprite«.

Versuchen Sie, den Programmablauf zunächst anhand des Ablaufplanes zu verstehen. Geben Sie dann das Programm ein. Speichern Sie das Programm, ehe Sie es starten. Durch die Betätigung einer beliebigen Taste wird das Programm beendet.

```
1000 rem *** ein einfaches sprite ***      <045>
1010 :                                     <224>
1020 :                                     <234>
1030 rem ** ausgabe programmeldung         <013>
1040 :                                     <000>
1050 print "{clr,8down}bitte warten";      <152>
1060 :                                     <020>
1070 :                                     <030>
1080 rem ** sprite-daten einlesen          <073>
1090 :                                     <050>
1100 for n = 896 to 958                    <018>
1110 :   poke n,255                        <198>
1120 next n                                <228>
1130 :                                     <090>
1140 :                                     <100>
1150 rem ** sprite-steuerung               <171>
1160 :                                     <120>
1170 print "{clr}";     :rem bilds.loeschen <113>
1180 poke 2040,14  :rem datenzeiger        <111>
1190 :                                     <150>
1200 vic = 53248   :rem vic-baustein       <045>
1210 poke vic,170  :rem horizont. pos.     <122>
1220 poke vic+1,120:rem vertikale pos.     <248>
1230 poke vic+39,13:rem gruenes sprite     <197>
1240 poke vic+21,1 :rem sprite 0 ein       <003>
1250 :                                     <210>
1260 :                                     <220>
1270 rem ** prog.ende nach tastendruck     <045>
1280 :                                     <242>
1290 get kp$                               <125>
1300 if kp$ = "" then 1290                 <075>
1310 :                                     <016>
1320 :                                     <026>
1330 rem ** register zuruecksetzen         <073>
1340 :                                     <046>
1350 poke vic+21,0 :rem ruecksetzung       <021>
1360 poke vic+39,0 :rem in umgekehrter     <119>
1370 poke vic+1,0  :rem reihenfolge...     <247>
1380 poke vic,0                            <157>
1390 :                                     <096>
1400 end                                   <132>
```

Listing 1. Das Programm »Ein einfaches Sprite«

Der erste Programmteil besteht aus nur einer Zeile:

```
1050 PRINT CHR$(147):PRINT "BITTE WARTEN";
```

Dieser Basic-Befehl löscht den Bildschirm und druckt die Programmeldung »BITTE WARTEN«, um dem Benutzer anzuzeigen, daß der C 64 mit der Arbeit begonnen hat, denn nichts ist beunruhigender als ein Programm, das nach dem Start keinerlei Aktivitäten zeigt.

Im zweiten Teil des Programms werden die 63 Sprite-Codes eingelesen:

```
1100 FOR N = 896 TO 958
1110 :   POKE N, 255
1120 NEXT N
```

Für dieses erste Programm wurde das einfachste Sprite ausgewählt, das möglich ist: alle Bildpunkte werden angezeigt. Daher haben alle 63 Codes den Wert 255. Der zweite Programmteil, die Schleife von 1100 bis 1120, schreibt diesen Wert nacheinander in die Speicherzellen 896 bis 958.

In Teil 3 des Programms wird die Hauptarbeit geleistet. Betrachten Sie zunächst die Zeilen 1170-1200:

```
1170 PRINT CHR$(147);   :REM BILDS.LOESCHEN
1180 POKE 2040,14       :REM DATENZEIGER
1190 :
1200 VIC = 53248        :REM VIC-BAUSTEIN
```

Zeile 1170 löscht den Bildschirm. In Zeile 1180 wird dem Rechner dann mitgeteilt, wo die Sprite-Daten zu finden sind, nämlich in den Speicherzellen 896 bis 958.

Wie ist das möglich? Der C 64 kann maximal acht Sprites, die von 0 bis 7 numeriert sind, gleichzeitig auf dem Bildschirm darstellen. Erhält er zum Beispiel den Befehl, Sprite 0 anzuzeigen, so berechnet er die Adresse, ab der die Daten für Sprite 0 abgelegt sind, indem er den Wert aus Speicherstelle 2040 mit 64 multipliziert. Da das Programm in Zeile 1180 den Wert 14 in diese Zelle geschrieben hat, ergibt sich aus 14 x 64 = 896 die Anfangsadresse der Sprite-Daten.

In Zeile 1200 erfolgt die Zuweisung des Wertes 53248 an die Variable VIC. Um welche Variable handelt es sich dabei? Hierzu müssen wir etwas weiter ausholen.

Die erstaunlichen Grafikfähigkeiten des C 64 beruhen auf einem speziell entwickelten Grafikbaustein, dem 6567-VIC.II (Video Interface Chip II, eine Weiterentwicklung des 6560-VIC aus dem Computer VC 20), kurz VIC.II genannt. Der VIC.II steuert die Textausgabe (40 Zeichen in 25 Zeilen) ebenso wie die hochauflösende Grafik (200 Zeilen zu 320 Bildpunkten) und die Darstellung der acht Sprites. Nur wer noch die Anfänge der Computergrafik auf Heimcomputern miterlebt hat, kann ermessen, welchen Fortschritt der VIC.II darstellt.

Zur Steuerung des VIC.II enthält dieser 47 adressierbare Speicherzellen, auch Register genannt. Die Anfangsadresse ist 53248, die Endadresse 53294. Nähere Angaben über die Register finden Sie im Anhang Ihres C 64-Handbuchs.

Zeile 1200 weist also der Variablen VIC den Wert 53248 zu. Damit können Sie jedes der 47 VIC-Register einfach adressieren, indem Sie die Registernummer (0-46) zu dem Wert von VIC addieren. Dies geschieht in den Zeilen 1210-1240:

```
1210 POKE VIC,170    :REM HORIZONT. POS.
1220 POKE VIC+1,120  :REM VERITKALE POS.
1230 POKE VIC+39,13  :REM GRUENES SPRITE
1240 POKE VIC+21,1   :REM SPRITE 0 EIN
```

Register 0 bestimmt die horizontale Position von Sprite 0. In Zeile 1210 wird dieser Wert auf 170 gesetzt. Die vertikale Position des Sprites steht in Register 1. Es erhält in Zeile 1220 den Wert 120. Der Sprite-Anfang liegt damit nahe der Bildschirmmitte. Register 39 definiert die Farbe aller darzustellenden Bildpunkte von Sprite 0 und erhält den Wert 13 für Hellgrün (eine Liste der verfügbaren Farben und ihrer Codes finden Sie ebenfalls im Anhang des Handbuches).

Sie haben nun die Sprite-Daten eingelesen, dem C 64 mitgeteilt, wo er diese Daten finden kann, die Sprite-Position zugeordnet und die Farbe der Bildpunkte festgelegt. Es fehlt nur noch die Anweisung zur Darstellung des Sprites. Diese Anweisung steht in Zeile 1240. Register 21 steuert die Anzeige der Sprites. Sobald es den Wert 1 erhält, wird Sprite 0 auf dem Bildschirm angezeigt.

Nun zum vierten Programmteil:

```
1290 GET KP$
1300 IF KP$ = "" THEN 1290
```

Zeile 1290 überprüft die Tastatur. In Zeile 1300 wird getestet, ob eine Taste betätigt worden ist. Wurde kein Zeichen eingegeben, verzweigt das Programm wieder nach Zeile 1290 und überprüft die Tastatur erneut. Wird schließlich eine Taste gedrückt, so beginnt das Programm mit dem fünften und letzten Programmteil.

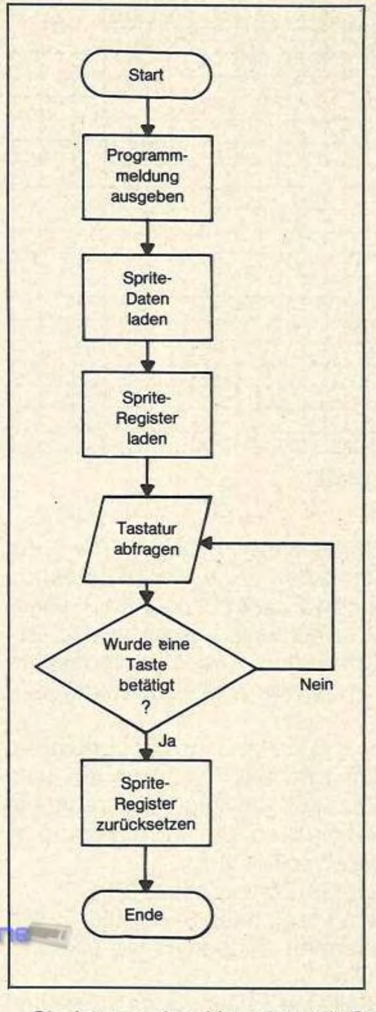


Bild 8. Zum besseren Verständnis: Der Ablaufplan für das Programm »Ein einfaches Sprite«

Es ist allgemein angebracht, Dinge so zu hinterlassen, wie man sie angetroffen hat, insbesondere bei der Programmierung von Computern. In den Zeilen 1350-1380 werden die VIC-Register wieder auf 0 zurückgesetzt:

```
1350 POKE VIC+21,0
:REM RUECKSETZUNG
1360 POKE VIC+39,0
:REM IN UMGEKEHRTER
1370 POKE VIC+1,0
:REM REIHENFOLGE...
1380 POKE VIC,0
```

Beachten Sie, daß die Register, im Vergleich zur Wertzuweisung in den Zeilen 1210-1240, in umgekehrter Reihenfolge zurückgesetzt werden.

Sie lernen den Umgang mit Sprites am besten, indem Sie bereits vorhandene Programme testweise modifizieren und dann verfolgen, welche Änderungen auftreten. Hier sind einige Vorschläge für die Modifikation des Programms »Ein einfaches Sprite«:

- Ändern Sie den Sprite-Code in Zeile 1110.
- Ändern Sie die Positionsangaben in Zeile 1210 und 1220.
- Ändern Sie den Farbcode in Zeile 1230.

Bei allen Experimenten sollten Sie niemals das Original-Programm »Ein einfaches Sprite« auf der Kassette oder Diskette überschreiben, denn Sie werden es in den folgenden Abschnitten noch als Rahmenprogramm für weitere Versuche mit Sprites benötigen. Ansonsten können und sollen Sie mit diesem Programm ruhig alle Experimente anstellen, die Ihnen im Zusammenhang mit Sprites gerade in den Sinn kommen. Arbeiten Sie dabei ruhig mit Variablen und Programmschleifen. Beispielsweise könnten Sie in Zeile 1230 für die Sprite-Farbe die Variable F verwenden, die Sie vorher per INPUT einlesen:

»1230 INPUT F:POKE VIC+39,F«

Die Positionierung von Sprites

Bei der Positionierung von Sprites teilen Sie dem Computer mit, an welcher Bildschirmposition der linke obere Bildpunkt des Sprites angezeigt werden soll. Der C 64 kann 200 Zeilen mit 320 Bildpunkten darstellen. Mit Hilfe der VIC-Register ist jedoch eine horizontale Positionierung bis zum Wert 512 und eine vertikale Positionierung bis zum Wert 256 möglich. Auf diese Weise können Sie Sprites in den Bildschirm hinein- oder herausgleiten lassen.

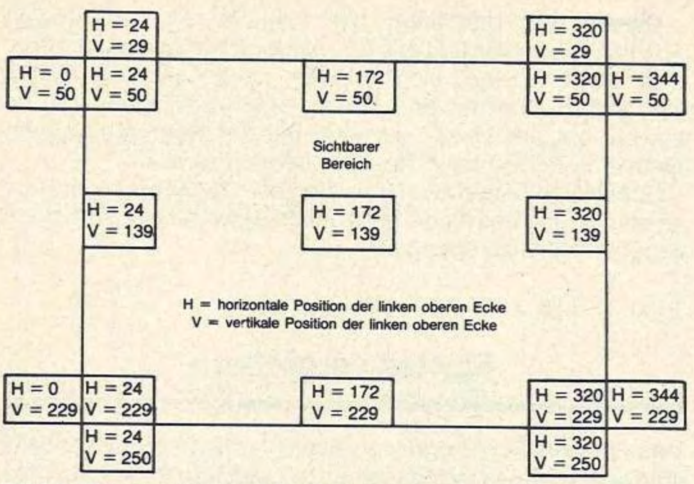


Bild 9. Einige wichtige horizontale und vertikale Bildschirmpositionen für Sprites in Normalgröße

In Bild 9 sind die Koordinaten einiger Randpositionen dargestellt. Sprites mit vertikalen Positionswerten kleiner als 30 liegen oberhalb des sichtbaren Bereichs. Sprites mit horizontalen Positionswerten zwischen 24 und 320 liegen dagegen immer innerhalb des Bildschirmbereichs.

Ein Sprite-JoJo

Laden Sie bitte das Programm »Ein einfaches Sprite« (Listing 1) und geben Sie dann die Zeilen aus Listing 2 ein. Dadurch werden einige Zeilen des Originalprogramms überschrieben und andere hinzugefügt. Benutzen Sie bitte genau die angegebenen Zeilennummern. Speichern und starten Sie dann das neue Programm.

```
1000 rem *** ein sprite-jojo                <176>
1220 poke vic+1,80   :rem vertikale pos     <026>
1251 :                                      <211>
1252 rem runter, dann hoch                  <143>
1253 :          0                           <213>
1254 for vp = 80 to 200                     <112>
1255 :    poke vic+1,vp                     <170>
1256 next vp                                <177>
1257 :                                      <217>
1258 for vp = 199 to 81 step -1             <051>
1259 :    poke vic+1,vp                     <174>
1260 next vp                                <181>
1261 :                                      <221>
1262 :                                      <222>
1300 if kp$ = "" then 1254                  <139>
```

Listing 2. Änderungen und Ergänzungen zur Umformung des Programms »Ein einfaches Sprite« in das Programm »Ein Sprite-JoJo«

Das Sprite bewegt sich wie ein JoJo auf und ab. Dieser Effekt wird durch die Zeilen 1254-1256 bewirkt:

```
1254 FOR VP = 80 TO 200
1255 :    POKE VIC+1,VP
1256 NEXT VP
```

In dieser Schleife durchläuft die vertikale Position die Werte 80 bis 200, wodurch sich das Sprite nach unten bewegt. In der anschließenden Schleife 1258-1260 wird diese Bewegung durch die vertikalen Positionswerte 199 bis 81 dann wieder umgekehrt:

```
1258 FOR VP = 199 TO 81 STEP -1
1259 :    POKE VIC+1,VP
1260 NEXT VP
```

Das Sprite bewegt sich wieder nach oben. Die neue Zeile 1300 bewirkt, daß dieses Auf und Ab bis zur Betätigung einer beliebigen Taste fortgesetzt wird.

```
1300 IF KP$ = "" THEN 1254
```

Die Definition von 512 horizontalen Positionen

Sicherlich haben Sie sich bereits gefragt, wie man eine horizontale Position größer als 255 definieren kann, obwohl ein Register nur Werte von 0 bis 255 aufnimmt.

Die Lösung ist einfach. Der VIC.II benutzt zur Festlegung der horizontalen Position eines Sprites nämlich nicht nur ein, sondern zwei Register, wobei das zweite allerdings kein vollwertiges Register ist, da es nur die Werte 0 und 1 aufnehmen kann. Der Wert 1 bedeutet, daß eine horizontale Position größer als 255 angesteuert werden soll. In diesem Fall addiert der C 64 den Wert 256 zum aktuellen Wert des ersten Registers. Enthält also zum Beispiel das erste Register den Wert 33 und das zweite Register den Wert 1, so ergibt sich für die anzusteuernde horizontale Position der Wert 298 (265 + 33). Enthält das zweite Register den Wert 0, so gilt nur der Wert des ersten Registers. Tabelle 1 zeigt einige Beispiele für horizontale Positionen zwischen 0 und 511.

1. Horizontales Register	2. Horizontales Register	Sprite Position
0	0	0
24	0	24
125	0	125
255	0	255
0	1	256
20	1	276
64	1	320
88	1	344
255	1	511

Tabelle 1. Einige Werte der horizontalen Positionsregister für Sprite-Positionen zwischen 0 und 511

Horizontale Bewegungen

Laden Sie bitte das Programm »Ein einfaches Sprite« (Listing 1), und fügen Sie die Zeilen aus Listing 3 hinzu. Speichern und starten Sie dann das neue Programm.

```
1000 rem *** horizontale bewegung ***          <172>
1210 poke vic,64    :rem horizont. pos.        <095>
1220 poke vic+1,139:rem vertikale pos.         <068>
1251 :                                         <211>
1252 rem rechts, dann links                    <015>
1253 :                                         <213>
1254 for hp = 64 to 280 step 2                 <129>
1255 :    sf = (hp > 255)                      <245>
1256 :    poke vic,hp + (sf * 256)             <097>
1257 :    poke vic+16, sf * (-1)               <138>
1258 next hp                                   <067>
1259 :                                         <219>
1260 for hp = 278 to 66 step-2                 <099>
1261 :    sf = (hp > 255)                      <251>
1262 :    poke vic,hp + (sf * 256)             <103>
1263 :    poke vic+16, sf * (-1)               <144>
1264 next hp                                   <073>
1265 :                                         <225>
1266 :                                         <226>
1300 if kp$ = "" then 1254                     <139>
```

Listing 3. Änderungen und Ergänzungen zur Umwandlung des Programms »ein einfaches Sprite« in das Programm »Horizontale Bewegung«

64er-online.de 64er-online.net

Ehe wir uns näher mit der Funktion der neuen Programmzeilen befassen, wollen wir noch einen kleinen Abstecher in die Welt der Logik unternehmen.

Bevor Sie ein Sprite an eine neue horizontale Position bewegen, müssen Sie immer folgende Aussage mit »wahr« oder »falsch« bewerten: Der neue Positionswert ist größer als 255. Von Ihrer Antwort hängt es ab, welcher Wert in das zweite horizontale Positionsregister geschrieben wird.

Das Commodore-Basic bewertet wahre Aussagen mit dem Wert -1 und falsche Aussagen mit dem Wert 0. Dieser Wahrheitswert kann einer Variablen zugeordnet werden. Dazu ein Beispiel:

```
100 LET WW = (5 >3)
```

Da 5 größer ist als 3, ist die Aussage (5 > 3) wahr, und WW erhält den Wert -1. Ein weiteres Beispiel:

```
100 LET WW = (36 = 21)
```

In diesem Fall erhält WW den Wert 0, denn die Aussage (36 = 21) ist falsch.

Das Sprite soll sich nun nach rechts bewegen. Durch eine kleine Änderung der Zeilen 1210-1220 beginnt diese Bewegung an einer neuen Position:

```
1210 POKE VIC,64    :REM HORIZONT. POS.
1220 POKE VIC+1,139:REM VERTIKALE POS.
```

Betrachten Sie nun bitte die Zeilen 1254-1258:

```
1254 FOR HP = 64 TO 280 STEP 2
1255 :    SF = (HP >255)
1256 :    POKE VIC,HP + (SF * 256)
1257 :    POKE VIC+16, SF * (-1)
1258 NEXT HP
```

Die Zeilen 1254-1258 bilden eine Schleife, in der die horizontale Position des Sprites in Zweierschritten die Werte von 64 bis 280 durchläuft. Bei jedem Schleifendurchgang wird dabei in Zeile 1255 geprüft,ob die neue Position den Wert 255 überschritten hat. Jeweils nach dem Wahrheitswert der überprüften Aussage in Zeile 1255 erfolgt dann die Positionierung in Zeile 1256 durch Eintragung der entsprechenden Werte in die beiden horizontalen Register.

Hat zum Beispiel HP den Wert 125, dann erhält der Größenfaktor SF in Zeile 1255 den Wert 0. Somit schreibt Zeile 1256 den Wert 125 +(0*256), also die Zahl 125, in das erste horizontale Register des Sprites. Das zweite horizontale Register erhält in Zeile 1257 den Wert -1*0=0. Beide Register enthalten damit die richtigen Werte für eine horizontale Position, die kleiner ist als 256.

Nun zu einer Position größer als 256. Hat HP beispielsweise den Wert 276, dann erhält SF in Zeile 1255 den Wert -1. Somit schreibt Zeile 1256 die Zahl 276+(-1*256) = 276-256 = 20 in das erste horizontale Register. Das zweite horizontale Register erhält in Zeile 1257 den Wert -1*-1=1. Wiederum enthalten beide Positionsregister somit die richtigen Werte.

Sollte Ihnen die Bedeutung der Zeilen 1245-1258 noch nicht ganz klar sein, so lesen Sie bitte noch einmal die letzten beiden Abschnitte, und überprüfen Sie die Formeln anhand einiger Werte aus Tabelle 1.

Betrachten Sie nun die Zeilen 1260-1264.

```
1260 FOR HP = 278 TO 66 STEP -2
1261 :    SF = (HP >255)
1262 :    POKE VIC, HP + (SF * 256)
1263 :    POKE VIC+16, SF * (-1)
1264 NEXT HP
```

Dieses Mal durchläuft die Schleife die horizontalen Positionswerte von 278 bis 66, wiederum in Zweierschritten. Das Sprite bewegt sich dadurch nach links. Die Zeilen 1261-1263 sind mit den Zeilen 1255-1257 identisch, da sowohl bei der Links- als auch bei der Rechtsbewegung dieselben horizontalen Register benutzt werden.

Schließlich haben Sie noch die Zeile 1300 verändert, um an den Anfang des für die horizontale Bewegung zuständigen Programmteils zu springen:

```
1300 IF KP$ = "" THEN 1254
```

Sprites vergrößern

Das Sprite in dem Programm »Ein einfaches Sprite« ist etwas langweilig. Um es interessanter zu gestalten, laden Sie bitte das Programm aus Listing 1 noch einmal und fügen die in Listing 4 dargestellten neuen Zeilen und Änderungen hinzu. Nach Abschluß dieser Arbeit speichern Sie das Programm (Sprite-Entwurf) dann wie üblich und starten es anschließend.

Anstelle des Quadrates erscheint jetzt die in Bild 7 entworfene und codierte Figur. Betrachten Sie dazu die neue Ladeschleife für die Sprite-Daten in den Zeilen 1100-1120:

```
1100 FOR N = 896 TO 958
1105 :     READ SPDTA
1110 :     POKE N, SPDTA
1120 NEXT N
```

Im alten Programm haben Sie in jede Speicherzelle den Wert 255 geschrieben, wodurch alle Bildpunkte sichtbar wurden. Nun benutzen Sie demgegenüber in Zeile 1005 eine READ-Anweisung, um die in DATA-Zeilen abgelegten Bildpunktdaten in die Variable SPDTA zu lesen und anschließend in den Speicher zu schreiben.

```
1000 rem *** sprite-entwurf ***                      <227>
1105 :   read spdta                                  <056>
1110 :   poke n, spdta                               <088>
1121 :                                               <081>
1122 data   0,  60,   0,   0,  36,   0               <104>
1123 data   0, 102,  24,   0, 102,  56               <195>
1124 data   0,  36,  56,   0,  60,  16               <223>
1125 data   0,  24,  16,   0,  24,  16               <189>
1126 data  15, 255, 240,   8, 126,   0               <118>
1127 data   8, 126,   0,   8,  24,   0               <037>
1128 data  28,  24,   0,  28,  24,   0               <196>
1129 data  24,  60,   0,   0,  60,   0               <020>
1130 data   0,  36,   0,   0,  36,   0               <145>
1131 data   0,  36,   0,   3, 231, 192               <216>
1132 data   3, 231, 192                              <138>
1230 poke vic+39,1  :rem weisse farbe                <242>
```

Listing 4. Änderung zum Programm »Sprite-Entwurf«

Betrachten Sie nun die elf DATA-Zeilen. Sie enthalten alle in Bild 7 berechneten Zahlencodes. Beachten Sie die Schreibweise der Daten: sie sind von links nach rechts, Zeile für Zeile, aus Bild 7 übernommen.

Schließlich wird durch die neue Version von Zeile 1230 das Sprite in weißer Farbe dargestellt, wodurch es besser sichtbar ist. Da sowohl die Technik eines Farbfernsehers als auch die des Commodore 64 nicht vollkommen ist, sind bestimmte Farben auf einigen Hintergrundfarben unterschiedlich gut zu erkennen. Sie müssen eine Weile experimentieren, um eine günstige Farbkombination zu finden.

Es gibt zwei Probleme im Zusammenhang mit dem letzten Programm. Erstens ist das Sprite zu klein, um in allen Details sichtbar zu sein, und zweitens ist dieser Entwurf nicht der Ihre.

Beginnen wir mit dem zweiten Problem. Zu Anfang dieses Kurses haben Sie mehrere Sprites entworfen und die 63 Zahlencodes für Ihren besten Entwurf berechnet. Benutzen Sie nun diese hart erarbeiteten Daten.

Laden Sie dazu das letzte Programm, »Sprite-Entwurf«, und lassen Sie sich die Zeilen 1122 bis 1132 ausgeben. Bewegen Sie dann den Cursor in die jeweilige Zeile und überschreiben Sie die alten Daten mit Ihren eigenen Bildpunktcodes.

Nun zum ersten Problem. Der VIC.II-Baustein ermöglicht sowohl eine horizontale als auch eine vertikale Vergrößerung von Sprites. In vergrößerten Sprites sind die Einzelheiten besser sichtbar. Fügen Sie daher dem Programm »Sprite-Entwurf« die folgenden Zeilen hinzu und speichern Sie es anschließend.

```
1000 REM *** EIN GROSSES SPRITE ***
1233 POKE VIC+23,1   :REM VERLAENGERN
1236 POKE VIC+29,1   :REM VERBREITERN
1353 POKE VIC+29,0
1356 POKE VIC+23,0
```

Sobald Sie das Programm gestartet haben, werden alle Einzelheiten Ihres Entwurfes sichtbar. Klopfen Sie sich auf die Schulter, ehe Sie sich näher mit der Vergrößerung von Sprites befassen.

Ein Sprite kann sowohl in doppelter Breite als auch in doppelter Höhe dargestellt werden. Das 30. VIC.II-Register mit der Adresse VIC+29 steuert die horizontale Vergrößerung aller acht Sprites. Der Wert 1 in diesem Register bewirkt, daß Sprite 0 in doppelter Breite ausgegeben wird. Enthält das Register den Wert 0, so hat Sprite 0 seine normale Breite.

Das 24. VIC.II-Register mit der Adresse VIC+23 steuert die vertikale Vergrößerung der acht Sprites. Wenn Sie eine 1 in dieses Register schreiben, verdoppelt sich die Höhe von Sprite 0. Enthält das Register eine 0, wird Sprite 0 in normaler Höhe ausgegeben.

Die Werte in den horizontalen und vertikalen Registern definieren auch für ein vergrößertes Sprite die Position seiner linken Ecke auf dem Bildschirm. Bild 10 zeigt einige wichtige Bildschirmpositionen für vergrößerte Sprites. Vergleichen Sie dieses Schema mit der Darstellung in Bild 9.

In den Zeilen 1233 und 1236 des Programms »Ein großes Sprite« wird der Wert 1 in beide Vergrößerungsregister geschrieben:

```
1233 POKE VIC+23,1: REM VERLAENGERN
1236 POKE VIC+29,1: REM VERBREITERN
```

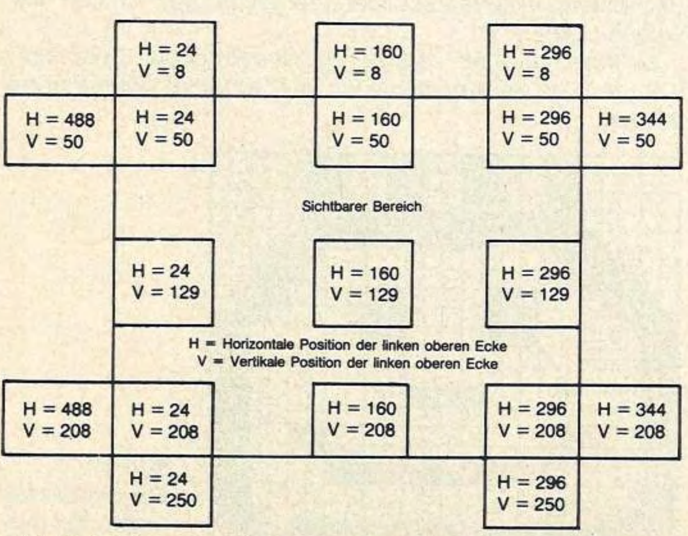


Bild 10. Einige Positionswerte für Sprites doppelter Größe

Dadurch vergrößert sich Sprite 0 in beide Richtungen. Am Ende des Programms, in den Zeilen 1353 und 1356, wird es durch den Wert 0 dann wieder auf seine Normalgröße zurückgesetzt:

```
1353 POKE VIC+29,0
1256 POKE VIC+23,0
```

Zusammenfassung

Im ersten Kapitel haben Sie gelernt:

- was Bildpunkte und was Sprites sind,
- wie Sie Ihr eigenes Sprite entwerfen und die zugehörigen 63 Zahlencodes berechnen können,
- wie die Sprite-Daten zu laden und die VIC.II-Register zu definieren sind, um ein einfaches Sprite auf dem Bildschirm darzustellen,
- wie die Position, Farbe und Größe eines Sprites anzugeben ist,
- wie ein Sprite in horizontaler und in vertikaler Richtung zu bewegen ist.

Nun haben Sie Gelegenheit, Ihr bisher erworbenes Wissen zu überprüfen. Vervollständigen Sie bitte die Sätze im Selbsttest und programmieren Sie die in den Programmierübungen geforderten Ergänzungen und Änderungen. Benutzen Sie dann die in diesem Kapitel vorgestellten Techniken und Hinweise, um eigene Programme zu schreiben. Nur Übung macht den Meister!

Selbsttest

Die Antworten zu diesem Selbsttest finden Sie im Anschluß an den Übungsteil.

1. Ein Sprite besteht aus einem Muster von 504 ..... , das über den Bildschirm bewegt werden kann.
2. Zur Codierung eines Sprite-Musters wird jede der 21 Zeilen in drei Gruppen zu je ..... Bildpunkten aufgeteilt.
3. Um ein Sprite sichtbar zu machen, müssen Sie bestimmte Werte in ein ..... des ..... -Bausteins schreiben.
4. Die Positionsangabe für ein Sprite bestimmt genaugenommen nur die Position der ..... Ecke des Sprite-Musters
5. Durch eine Veränderung des ..... Positionswertes bewegt sich ein Sprite auf oder ab.
6. Um die horizontale Position eines Sprites festzulegen, müssen Sie .... Positionsregister benutzen, da es ..... horizontale Positionen gibt.
7. Durch die folgende Basic-Anweisung erhält die Variable TV den Wert ..... :
   `10 LET TV = (17 < 5)`
8. Modifizieren Sie die Zeile 1230 des Programms »Sprite-Entwurf« derart, daß das Sprite in gelber Farbe dargestellt wird.
   `1230 .....`
9. Ein Sprite kann in ..... und in ..... Richtung oder in beiden Richtungen vergrößert werden.

Programmierübungen

Alle Aufgaben können durch Erweiterung oder Änderung des Programms »Ein einfaches Sprite« (Listing 1) gelöst werden, aber auch durch Neuprogrammierung. Selbstverständlich stellt auch jedes andere lauffähige Programm, daß den gleichen Zweck erfüllt, eine Lösung dar.

1. Veranlassen Sie das Sprite, sich auf einer rechteckigen Bahn zu bewegen.
2. Verändern Sie das Programm so, das sich hin und wieder die Farbe des Sprites ändert.
3. Schreiben Sie eine Programmschleife, die das Sprite nacheinander in vier Größen darstellt: normal, verbreitert, verlängert und in beide Richtungen vergrößert.

Antworten zum Selbsttest

Hier sind die wohl zutreffendsten Antworten. Aber sicherlich sind auch andere sinnvolle Antworten möglich.

1. Bildpunkte
2. acht
3. Register, VIC.II
4. linken oberen
5. vertikalen
6. zwei, 512
7. null
8. 1230 POKE VIC+39,7 : REM GELBE FARBE
9. horizontaler, vertikaler (vertikaler, horizontaler)

Lösungsvorschläge für die Programmierübungen

Die folgenden drei Lösungsvorschläge basieren auf dem Programm »Ein einfaches Sprite« aus Listing 1. Dargestellt sind die Zeilen, die geändert oder hinzugefügt werden müssen, um die Aufgabe zu erfüllen.

64ER ONLINE

1. Laden Sie das Programm »Ein einfaches Sprite« und geben Sie die folgenden Zeilen ein (Programm »Rechteckbewegung«):

```
1000 REM *** RECHTECKBEWEGUNG ***
1210 POKE VIC,82     :REM HORIZONT. POS.
1220 POKE VIC+1,100 :REM VERTIKALE POS.
1243 REM ** BEWEGUNG NACH RECHTS, UNTEN,
1244 REM    LINKS, DANN AUSGANGSPUNKT
1246 REM    (HORIZONTALE DREIERSPRUENGE
1247 REM    ZUR ANPASS. AN VERT.GESCH.)
1249 FOR HP = 84 TO 261 STEP 3 :REM ->  R
1250 :   SF = (HP >  R 255)
1251 :   POKE VIC, HP + (SF * 256)
1252 :   POKE VIC+16, SF * (-1)
1253 NEXT HP
1255 FOR VP = 101 TO 179      :REM RUNTER
1256 :   POKE VIC+1, VP
1257 NEXT VP
1259 FOR HP = 258 TO 87 STEP -3 :REM <E-
1260 :   SF = (HP >  R 255)
1261 :   POKE VIC, HP + (SF * 256)
1262 :   POKE VIC+16, SF * (-1)
1263 NEXT HP
1265 FOR VP = 178 TO 100 STEP -1 : REM
1266 : POKE VIC+1, VP
1267 NEXT VP
1300 IF KP$ = "" THEN 1249
```

2. Laden Sie das Programm »Ein einfaches Sprite« und geben Sie die folgenden Zeilen ein (Programm »Farbwechsel«):

```
1000 REM *** FARBWECHSEL ***
1071 REM ** ANFANGSFARBE FESTLEGEN
1073 OC = 13        :REM START MIT GRUEN
1230 POKE VIC+39,OC :REM ANFANGSFARBE
1252 REM ** FARBEN WECHSELN
1254 FOR DELAY = 1 TO 500 : NEXT
1255 NC µ  0 OC + 1           :REM FARBE NEU
1256 IF NC=16 THEN NC=0  :REM FARBE 1-15
1257 POKE VIC+39,NC      :REM ALTE FARBE
1258 OC = NC             :REM NEUE FARBE
1300 IF KP$ = "" THEN 1254
```

3. Laden Sie das Programm »Ein einfaches Sprite« und geben Sie die folgenden Zeilen ein (Programm »Größenwechsel«):

```
1000 REM *** GROESSENWECHSEL ***
1252 REM ** HORIZONTALE VERGROESSERUNG
1254 FOR PAUSE = 1 TO 400   : NEXT
1255 POKE VIC+29,1
1258 REM ** HORIZONTAL VERKLEINERN UND
1259 REM    VERTIKAL VERGROESSERN
1261 FOR PAUSE = 1 TO 400   : NEXT
1262 POKE VIC+29,0
1263 POKE VIC+23,1
1266 REM ** HORIZONTALE VERGROESSERUNG
1268 FOR PAUSE = 1 TO 400   : NEXT
1269 POKE VIC+29,1
1272 REM ** HORIZONTAL VERKLEINERN UND
1273 REM    VERTIKAL VERKLEINERN
1275 FOR PAUSE = 1 TO 400   : NEXT
1276 POKE VIC+29,0
1277 POKE VIC+23,0
1280 REM ** PROGR.ENDE MIT TASTENDRUCK
1282 IF KP$ = "" THEN 1254
```

Kapitel 2: Mehrere Sprites

In diesem Kapitel lernen Sie, mehrere Sprites gleichzeitig auf dem Bildschirm darzustellen, aus den Daten eines Sprites mehrere identische oder unterschiedlich große Sprites zu erzeugen und zwei Sprites gleichzeitig über den Bildschirm gleiten zu lassen.

Sprite-Kopien

Der C 64 erlaubt die Darstellung mehrerer Sprites, auch wenn nur ein Satz von Sprite-Codes vorhanden ist. Wenn Sie ein solches Sprite in gleicher Größe an unterschiedlichen Bildschirmpositionen abbilden, sehen Sie vier Kopien desselben Sprites.

Listing 5 zeigt ein entsprechendes Programm, »Einfache Kopien«, das vier Kopien des in Bild 11 dargestellten Sprites erzeugt.

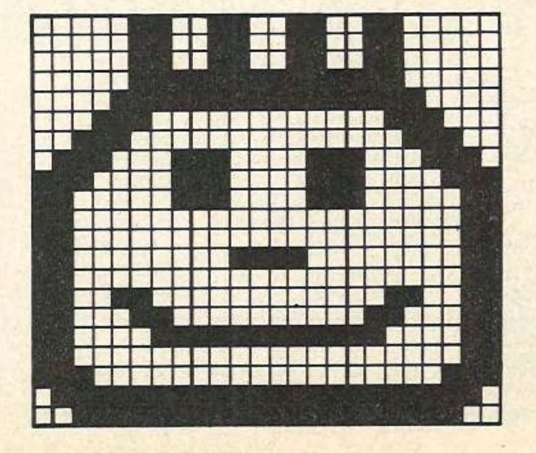


Bild 11. Ein einfaches Sprite für den Kopiervorgang

Das Programm entspricht in seinem Aufbau dem Programm »Ein einfaches Sprite« aus Kapitel 1, mit dem Unterschied, daß jetzt Datenanzeiger, Positionsangaben und Farben für vier Sprites vorhanden sind. Geben Sie das Programm ein, speichern und starten Sie es.

In Kapitel 1 haben Sie gelernt, daß Adresse 2040 dazu dient, dem Rechner mitzuteilen, an welcher Speicherstelle sich die Pixelcodes für Sprite 0 befinden. Die entsprechenden Zeiger für die Sprites 1-7 stehen in den Adressen 2041 bis 2047 (Tabelle 2).

Adresse →	2040	2041	2042	2043	2044	2045	2046	2047
Zeigt auf Pixel-Daten für Sprite-Nummer →	0	1	2	3	4	5	6	7

Tabelle 2. Die Speicherzellen der Sprite-Datenzeiger

Betrachten wir die wichtigen Abschnitte des Programms (Listing 8) einmal genauer. Die Zeilen 1000-1310 kommen Ihnen sicherlich bekannt vor. Die Rückmeldung des Programms wird ausgegeben, die Bildpunktdaten werden in die Speicherzellen 896-958 gelesen, der Bildschirm wird gelöscht, und die Variable VIC erhält die Anfangsadresse des VIC.II-Bausteins.

Der erste Unterschied zum alten Programm liegt in den Zeilen 1330 bis 1360:

```
1330 POKE 2040,14    :REM DATENZEIGER 0
1340 POKE 2041,14    :REM DATENZEIGER 1
1350 POKE 2042,14    :REM DATENZEIGER 2
1360 POKE 2043,14    :REM DATENZEIGER 3
```

Statt eines Sprites sind vier Sprites darzustellen, wobei der Rechner jedesmal dieselben 63 Werte aus den Speicherzellen ab Adresse (14*64)=896 lesen soll. In den Zeilen 1380-1460 wird jedem Sprite eine horizontale und eine vertikale Bildschirmposition zugeordnet:

```
1380 POKE VIC,98      :REM HORZNTL.POS. 0
1390 POKE VIC+2,246 :REM HORZNTL.POS. 1
1400 POKE VIC+4,98   :REM HORZNTL.POS. 1
1410 POKE VIC+6,246 :REM HORZNTL.POS. 3
1420 :
1430 POKE VIC+1,95   :REM VERTKAL.POS. 0
1440 POKE VIC+3,95   :REM VERTKAL.POS. 1
1450 POKE VIC+5,184 :REM VERTCAL.POS. 2
1460 POKE VIC+7,184 :REM VERTCAL.POS. 3
```

Die Adresse VIC (53248) ist das Register für die horizontale Position von Sprite 0, und VIC+1 (53249) das Register für die vertikale Position dieses Sprites. Entsprechend sind die anderen 14 Register den Sprites 1 bis 7 zugeordnet. VIC+2 (53250) ist das horizontale und VIC+3 (53251) das vertikale Register für Sprite 1 und am Ende steht VIC+15 (53263), das vertikale Register für Sprite 7.

Als aufmerksamer Leser werden Sie jetzt fragen: Wo bleibt das zweite horizontale Register für jedes Sprite? Wie Sie aus Kapitel 1 wissen, können diese Register nur den Wert 0 oder 1 enthalten. Alle acht finden daher in einer Speicherstelle Platz, in der Adresse VIC+16 (53264). Sie werden diese Register später genauer kennenlernen.

Farben zuordnen und Sprites anzeigen

Die Zuordnung der Farben erfolgt in den Zeilen 1480-1510:

```
1000 rem *** einfache kopien ***                 <209>
1010 :                                           <224>
1020 :                                           <234>
1030 rem ** ausgabe programmeldung               <013>
1040 :                                           <000>
1050 print "{clr,8down}bitte warten"             <034>
1060 :                                           <020>
1070 :                                           <030>
1080 rem ** sprite-daten laden                   <212>
1090 :                                           <050>
1100 for n = 896 to 958                          <018>
1110 :   read spdta                              <061>
1120 :   poke n, spdta                           <098>
1130 next n                                      <238>
1140 :                                           <100>
1150 data   6,102, 96,  6,102, 96                <148>
1160 data   6,102, 96,  7,255,224                <122>
1170 data  15,255,240, 28,  0, 56                <157>
1180 data  56,  0, 28,113,195,142                <193>
1190 data 225,195,135,193,195,131                <090>
1200 data 192,  0,  3,192,  0,  3                <164>
1210 data 192, 60,  3,192,  0,  3                <061>
1220 data 204,  0, 51,198,  0, 99                <101>
1230 data 195,255,195,192,  0,  3                <116>
1240 data 224,  0,  7,127,255,254                <004>
1250 data  63,255,252                            <168>
1260 :                                           <220>
1270 :                                           <230>
1280 rem ** sprite-register vorbereiten          <000>
1290 :                                           <252>
1300 print "{clr}"        :rem schirm loesch
     .                                           <079>
1310 vic = 53248      :rem grafikbaustein        <125>
1320 :                                           <026>
1330 poke 2040,14     :rem datenzeiger 0         <031>
1340 poke 2041,14     :rem datenzeiger 1         <233>
1350 poke 2042,14     :rem datenzeiger 2         <180>
1360 poke 2043,14     :rem datenzeiger 3         <126>
1370 :                                           <076>
1380 poke vic,98      :rem horzntl.pos. 0        <111>
1390 poke vic+2,246 :rem horzntl.pos. 1          <061>
1400 poke vic+4,98    :rem horzntl.pos. 2        <065>
1410 poke vic+6,246 :rem horzntl.pos. 3          <091>
1420 :                                           <126>
1430 poke vic+1,95    :rem vertkal.pos. 0        <180>
1440 poke vic+3,95    :rem vertkal.pos. 1        <193>
1450 poke vic+5,184 :rem vertkal.pos. 2          <202>
1460 poke vic+7,184 :rem vertkal.pos. 3          <217>
1470 :                                           <176>
1480 poke vic+39,1    :rem 0 ist weiss           <064>
1490 poke vic+40,3    :rem 1 ist hellblau        <089>
1500 poke vic+41,5    :rem 2 ist gruen           <015>
1510 poke vic+42,7    :rem 3 ist gelb            <205>
1520 :                                           <226>
1530 poke vic+21,15 :rem sprites 0-3 an          <180>
1540 :                                           <248>
1550 :                                           <002>
1560 rem ** prog.ende mit tastendruck            <071>
1570 :                                           <022>
1580 get kp$                                     <161>
1590 if kp$ = "" then 1580                       <119>
1600 :                                           <052>
1620 rem ** ruecksetzen des registers            <207>
1630 :                                           <082>
1640 poke vic+21,0                               <140>
1650 :                                           <102>
1660 end                                         <138>
```

Listing 5. Ausdruck des Programms »Einfache Kopien«

```
1480 POKE VIC+39,1    :REM 0 IST WEISS
1490 POKE VIC+40,3    :REM 1 IST HELLBLAU
1500 POKE VIC+41,5    :REM 2 IST GRUEN
1510 POKE VIC+41,7    :REM 3 IST GELB
```

Wie Sie sicherlich vermutet haben, enthalten die acht aufeinanderfolgenden Register VIC+39 (53287) bis VIC+46 (53294) die Farbcodes für die einzelnen Sprites. Zeile 1530 sorgt schließlich für die Ausgabe der Sprites 0, 1, 2 und 3 auf dem Bildschirm:

```
1530 POKE VIC+21,15   :REM SPRITES 0-3 AN
```


64er-online.de
64er-online.net

Welcher Zusammenhang besteht aber zwischen der Zahl 15 und den Nummern 0, 1, 2 und 3 der Sprites? Zur Beantwortung dieser Frage ist ein Ausflug in die Welt der Bits und Bytes erforderlich.

Bits und Bytes

Vielleicht haben Sie sich schon bei der Einteilung der Bildpunktzeilen eines Sprites in Gruppen zu je acht Bildpunkten gefragt, warum es gerade acht Bildpunkte sein müssen und nicht neun, zehn oder 24.

Der Grund liegt in der Art und Weise, in der Computer wie der C 64 Zahlen verarbeiten. Der Rechner kennt eigentlich nur die beiden Werte 0 und 1. Ein Zahlensystem, in dem alle Zahlen nur durch diese beiden Werte ausgedrückt werden, nennt man Binärsystem.

Die beiden binären Zeichen 0 und 1 heißen »Bits«. Eine Gruppe von acht Bits bezeichnet man als ein »Byte«. Jede der Speicherzellen des C 64, auch die Register, können jeweils ein Byte aufnehmen. Die Bilder 12 und 13 zeigen die Darstellung einiger Bytes. Mit 8 Bit lassen sich die Zahlen von 0 bis 255 darstellen.

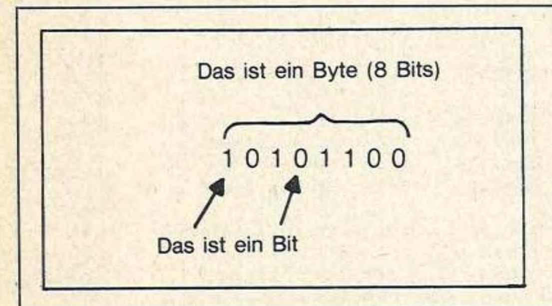


Bild 12. Ein Byte besteht aus 8 Bit

Viele der VIC.II-Register ermöglichen die gleichzeitige Steuerung aller acht Sprites, da jeweils eines der 8 Bit pro Register einem Sprite zugeordnet ist und somit eine Art Miniaturregister darstellt.

So ist zum Beispiel das Register VIC+21 (53269) das Zentralregister zur Steuerung der Sprite-Anzeige. Jedes Bit dieses Registers ist einem Sprite zugeordnet und bestimmt, ob das Sprite sichtbar ist oder nicht.

Zahl →	255	240	128	127	60	15	1	0
Byte-Schreibweise →	11111111	11110000	10000000	01111111	00111100	00001111	00000001	00000000

Bild 13. In einem Byte können die Dezimalwerte (Basis 10) von 0 bis 255 in binärer Form (8 Bit) dargestellt werden

Die 8 Bit eines Bytes werden von rechts nach links durchnumeriert, erhalten also die Nummern 0 bis 7. In Register VIC+21 steuert Bit 0 das Sprite 0, Bit 1 das Sprite 1 und so weiter. Um ein bestimmtes Sprite sichtbar werden zu lassen, genügt es, das entsprechende Bit auf den Wert 1 zu setzen. Um es wieder verschwinden zu lassen, setzen Sie das Bit wieder auf 0.

Um also die vier Sprites 0 bis 3 anzuzeigen, muß in das Anzeige-Zentralregister VIC+21 eine Zahl geschrieben werden, deren Binärwert viermal die 1 (für die Bits 0, 1, 2 und 3) und viermal die 0 (für die Bits 4 bis 7) enthält. Sie können dazu ebenfalls die Tabelle 1 benutzen.

Bild 14 zeigt ein Byte mit den numerierten Bits, ihren Codes sowie den beiden möglichen Bitwerten. Um den richtigen Zahlenwert zur Steuerung der Sprites zu erhalten, tragen Sie zunächst für jedes Sprite, das sichtbar werden soll, eine 1 an der entsprechenden Bitposition ein und für jedes Byte, das unsichtbar bleiben soll, eine 0. Die Addition der Codes aller Bits, deren Wert 1 ist, ergibt dann die gesuchte Zahl, deren binäre Darstellung die Reihe der eingetragenen 0/1-Werte ist.

Einige Beispiele dazu finden Sie in Bild 15. Die Tabelle enthält auch den im Programm »Einfache Kopien« benutzten Wert. Es sollen dort die Sprites 0-3 sichtbar gemacht werden. Dazu müssen Sie die Bits 0 bis 3 auf 1 setzen. Die Addition der entsprechenden Bitcodes ergibt (8 + 4 + 2 + 1) = 15. Damit ist das Geheimnis der Zahl 15 in Zeile 1530 gelüftet.

Bit-Code →	128	64	32	16	8	4	2	1
Ein beliebiges Byte →	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0	1 or 0
Bit-Nummer →	7	6	5	4	3	2	1	0

Bild 14. Ein Byte mit seinen numerierten 8 Bit, den Bit-Codes und den möglichen Bitwerten

Sichtbare Sprites	Unsichtbare Sprites	Registerbyte (Jedes Bit steuert das Sprite mit der gleichen Nummer)									Wert für den POKE-Befehl
–	0–7	Bit-Code	128	64	32	16	8	4	2	1	
			0	0	0	0	0	0	0	0	
		Bit-Nummer	7	6	5	4	3	2	1	0	0
0	1–7	Bc	128	64	32	16	8	4	2	1	1=
			0	0	0	0	0	0	0	1	1
		Bn	7	6	5	4	3	2	1	0	
0.1	2–7	Bc	128	64	32	16	8	4	2	1	1+2=
			0	0	0	0	0	0	1	1	3
		Bn	7	6	5	4	3	2	1	0	
0–3	4–7	Bc	128	64	32	16	8	4	2	1	1+2+4+8=
			0	0	0	0	1	1	1	1	15
		Bn	7	6	5	4	3	2	1	0	
0.2. 4.6	1.3. 5.7	Bc	128	64	32	16	8	4	2	1	1+4+16+ 64=
			0	1	0	1	0	1	0	1	
		Bn	7	6	5	4	3	2	1	0	85
0-7	–	Bc	128	64	32	16	8	4	2	1	1+2+4+8+ 16+32+64
			1	1	1	1	1	1	1	1	+128=
		Bn	7	6	5	4	3	2	1	0	255

Bild 15. Beispiel zur Steuerung der Sprites

Die restlichen Zeilen des Programms »Einfache Kopien« sind Ihnen bereits bekannt. In den Zeilen 1580-1590 wartet der Rechner auf einen Tastendruck. Sobald eine Taste betätigt wird, setzt Zeile 1640 die Sprite-Register zurück. Dabei zeigt sich eine kleine Besonderheit: nicht jedes Register muß zurückgesetzt werden.

Welche Register sollten Sie wieder auf 0 setzen? Zunächst das Zentralregister VIC+21, falls alle Sprites wieder verschwinden sollen. Wenn Sie ein Sprite verbreitert oder verlängert haben, ist es angebracht, auch die entsprechenden Register VIC+23 und VIC+29 wieder auf 0 zu setzen. Dadurch vermeiden Sie, daß ein Sprite ungewollt in vergrößerter Form dargestellt wird.

Den durch das vorangegangene Programm erzeugten vier Sprites lag zwar der gleiche Satz von Zahlencodes zugrunde, sie wurden jedoch in unterschiedlichen Farben dargestellt.

64er-online.de

Weitere Unterschiede lassen sich durch Verlängern oder Verbreitern erzeugen.

Modifizierte Sprite-Kopien

Wie Sie aus Kapitel 1 wissen, steuert das Register VIC+29 die Verbreiterung der Sprites. Jedem Bit des Registers ist ein Sprite zugeordnet. Um zum Beispiel die Sprites 2 und 3 zu verbreitern, müssen Sie Bit 2 und 3 des Registers auf 1 setzen. Mit Hilfe von Bild 14 finden Sie für die in das Register einzugebende Zahl den Wert (8 + 4) = 12. Dieser Wert ist in Tabelle 3 dargestellt.

Bit-Code →	128	64	32	16	8	4	2	1
Bit →	0	0	0	0	1	1	0	0
Bit-Nummer →	7	6	5	4	3	2	1	0

8 + 4 = 12

Tabelle 3. Die Eingabe des Wertes 12 in das horizontale Vergrößerungsregister setzt Bit 2 und 3 auf 1 und bewirkt die horizontale Vergrößerung der Sprites 2 und 3

Das Register VIC+23 steuert in gleicher Weise die Verlängerung von Sprites. Um beispielsweise Sprite 1 und 3 zu verlängern, müssen Sie Bit 1 und 3 auf 1 setzen. Der entsprechende Zahlenwert für das Register ist (8 + 2) = 10. Er ist in Tabelle 4 dargestellt.

Bit-Code →	128	64	32	16	8	4	2	1
Bit →	0	0	0	0	1	0	1	0
Bit-Nummer →	7	6	5	4	3	2	1	0

8 + 2 = 10

Tabelle 4. Die Eingabe des Wertes 10 in das vertikale Vergrößerungsregister setzt Bit 1 und 3 auf 1 und bewirkt die vertikale Vergrößerung der Sprites 1 und 3

Sie besitzen nun das nötige Rüstzeug, um das Programm »Einfache Kopien« zu modifizieren. Laden Sie es und fügen Sie die Zeilen aus Listing 9 hinzu. Damit erhalten Sie das Programm »Modifizierte Kopien«. Speichern und starten Sie das neue Programm.

Der Bildschirm zeigt nun vier Sprites, die alle aus derselben Datengruppe erzeugt werden und die sich doch alle voneinander unterscheiden. Die neuen Zeilen 1400, 1410, 1440 und 1460 verteilen die Sprites auf dem Bildschirm, und die beiden Zeilen 1522 und 1524 sorgen für eine Vergrößerung entsprechend den beiden genannten Beispielen:

```
1000 rem*** unterschiedliche kopien ***    <108>
1400 poke vic+4,86  :rem horzntl.pos. 2    <055>
1410 poke vic+6,243 :rem horzntl.pos. 3    <067>
1440 poke vic+3,85  :rem vertkal.pos. 1    <191>
1460 poke vic+7,174 :rem vertkal.pos. 3    <213>
1522 poke vic+23,10 :rem 1 & 3 schmal      <197>
1524 poke vic+29,12 :rem 2 & 3 breit       <046>
1526 :                                     <232>
1642 poke vic+23,0                         <144>
1644 poke vic+29,0                         <152>
```

Listing 6. Änderungen und Erweiterungen zur Umwandlung des Programms »Einfache Kopien« in das Programm »Unterschiedliche Kopien«

```
1522 POKE VIC+23,10  :REM 1 & 3 SCHMAL
1524 POKE VIC+29,12  :REM 2 & 3 BREIT
```

Sprite 0 bleibt unverändert. Sprite 1 wird verlängert und Sprite 2 verbreitert. Sprite 3 wird sowohl verlängert als auch verbreitert. Ein Tastendruck beendet das Programm und setzt das Vergrößerungsregister auf 0 zurück.

In den meisten Fällen werden Sie Sprites unterschiedlicher Struktur benötigen. Dazu müssen Sie für jedes Sprite eine eigene Datengruppe (Bildpunktcodes) laden, und es stellt sich die Frage, wo die jeweils 63 Zahlen am besten unterzubringen sind.

Wenn Sie nur drei oder weniger Sprites darstellen wollen, stehen folgende Speicherbereiche zur Verfügung: 832-894, 896-958, 960-1022. Dieser gesamte Bereich ist der Puffer für den Kassettenrekorder und wird während des Programmlaufes nicht benutzt. Falls Sie den Kassettenpuffer benutzen, müssen die Datenzeiger der entsprechenden Sprites (2040-2047) die Anfangsadresse des jeweiligen Datenblocks, geteilt durch 64, enthalten. Für den Kassettenpuffer sind dies die Werte 13, 14 und 15.

Für die Daten von mehr als drei Sprites empfiehlt sich der Speicherbereich ab Adresse 12288. Tabelle 5 zeigt die jeweiligen Anfangsadressen pro Block sowie die zugehörigen Werte für die Datenzeiger. Dem fortgeschrittenen Systemprogrammierer des C 64 stehen zwar noch weitere Speicherbereiche zur Verfügung, deren Manipulation ist jedoch nicht Thema dieses Buches.

63-Byte Speicherbereich →	12288 — 12350	12352 — 12414	12416 — 12478	12480 — 12542	12544 — 12606	12608 — 12670	12672 — 12734	12736 — 12798
Datenzeiger auf →	192	193	194	195	196	197	198	199

Tabelle 5. Die für die Speicherung von SpriteDaten geeigneten Bereiche und die entsprechenden Werte der Datenzeiger

Zwei unterschiedliche Sprites

Worin wird sich ein Programm, das zwei unterschiedliche Sprites auf dem Bildschirm darstellt, von dem Programm »Sprite-Entwurf« aus Kapitel 1 unterscheiden?

Erstens muß es statt einem zwei Datenblöcke laden, zweitens die Datenzeiger in Adresse 2040 und 2041 auf die Speicherbereiche mit den geladenen Daten richten, und drittens muß es die entsprechenden Werte für die Position, Größe und Farbe jedes Sprites in die VIC.II-Register schreiben. Schließlich sind noch beide Sprites sichtbar zu machen.

Bild 16 zeigt zwei neue Sprite-Entwürfe. Listing 7 enthält das Programm »Ein Sprite-Paar«, das die beiden Figuren auf dem Bildschirm darstellt. Geben Sie das Programm ein und speichern Sie es. Anschließend können Sie das Programm starten und Ihren Vorstellungen entsprechend modifizieren.

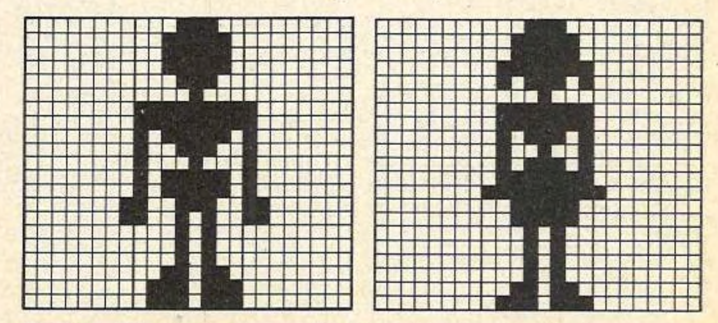


Bild 16. Die zwei neuen Sprites für »Ein Sprite-Paar«

64er-online.de 64er-online.net

```
1000 rem *** ein sprite-paar ***            <255>
1010 :                                      <224>
1020 :                                      <234>
1030 rem ** ausgabe programmeldung          <013>
1040 :                                      <000>
1050 print "{clr,8down}bitte warten";       <152>
1060 :                                      <020>
1070 :                                      <030>
1080 rem ** sprite-daten laden              <212>
1090 :                                      <050>
1100 for n = 896 to 958   :rem 1.sprite     <124>
1105 :   read spdta                         <056>
1110 :   poke n, spdta                      <088>
1130 next n                                 <238>
1140 print". ";           :rem pktausgb     <121>
1150 :                                      <110>
1160 for n = 960 to 1022  :rem 2.sprite     <210>
1165 :   read spdta                         <116>
1170 :   poke n, spdta    0                 <148>
1190 next n                                 <042>
1200 print". ";           :rem pktausgb     <181>
1210 :                                      <170>
1220 data  0, 28,  0,  0, 62,  0            <252>
1230 data  0, 62,  0,  0, 62,  0            <004>
1240 data  0, 28,  0,  0,  8,  0            <088>
1250 data  0,255,128,  0,255,128            <109>
1260 data  0,190,128,  0,156,128            <087>
1270 data  0,136,128,  0,190,128            <218>
1280 data  0,190,128,  1,156,192            <060>
1290 data  1,148,192,  0, 20,  0            <243>
1300 data  0, 20,  0,  0, 20,  0            <204>
1310 data  0, 54,  0,  0,119,  0            <184>
1320 data  0,119,  0                        <029>
1330 :                                      <036>
1340 data  0, 28,  0,  0, 62,  0            <118>
1350 data  0, 62,  0,  0,127,  0            <030>
1360 data  0, 93,  0,  0,  8,  0            <113>
1370 data  0,127,  0,  0,127,  0            <090>
1380 data  0, 93,  0,  0, 73,  0            <109>
1390 data  0, 93,  0,  0,127,  0            <231>
1400 data  0,255,128,  0, 62,  0            <079>
1410 data  0, 62,  0,  0, 20,  0            <058>
1420 data  0, 20,  0,  0, 20,  0            <068>
1430 data  0, 20,  0,  0, 54,  0            <254>
1440 data  0,119,  0                        <149>
1450 :                                      <156>
1460 :                                      <166>
1470 rem ** sprite-register vorbereiten     <190>
1480 :                                      <186>
1490 print "{clr}"       :rem schirm loesch
     .                                      <013>
1500 vic = 53248     :rem grafikbaustein    <059>
1510 :                                      <216>
1520 poke 2040,14    :rem datenzeiger 0     <221>
1530 poke 2041,15    :rem datenzeiger 1     <169>
1540 :                                      <248>
1550 poke vic,124    :rem horzntl.pos. 0    <150>
1560 poke vic+2,173 :rem horzntl.pos. 1     <219>
1570 poke vic+1,150 :rem vertkal.pos. 0     <014>
1580 poke vic+3,150 :rem vertkal.pos. 1     <029>
1590 :                                      <042>
1600 poke vic+39,3  :rem 0 ist hellblau     <049>
1610 poke vic+40,7  :rem 1 ist gelb         <241>
1620 :                                      <072>
1630 poke vic+23,3  :rem beide sprites      <139>
1640 poke vic+29,3  :rem dopp. groesse      <165>
1650 :                                      <102>
1660 poke vic+21,3  :rem beide sichtb.      <184>
1670 :                                      <122>
1680 :                                      <132>
1690 rem ** prog.ende mit tastendruck       <201>
1700 :                                      <152>
1710 get kp$                                <035>
1720 if kp$ = "" then 1710                  <153>
1730 :                                      <182>
1740 :                                      <192>
1750 rem ** ruecksetzen des registers       <081>
1760 :                                      <212>
1770 poke vic+21,0  :rem sprites aus        <176>
1780 poke vic+29,0  :rem und wieder         <006>
1790 poke vic+23,0  :rem normalgroesse      <209>
1800 :                                      <254>
1810 end                                    <034>
```

Listing 7. Ausdruck des Programms »Ein Sprite-Paar«

Das Programm enthält keine neuen Anwendungen. Zeile 1050 löscht den Bildschirm und gibt die Programmeldung aus. Zwei Schleifen laden dann die jeweils 63 Daten der beiden Sprites. Der erste Datenblock wird in den Speicherbereich 896-958 geschrieben. Das Programm signalisiert den Abschluß dieses Ladevorgangs, indem es in Zeile 1140 einen Punkt hinter die Programmeldung setzt. Ein weiterer Punkt wird in Zeile 1200 ausgegeben, nachdem der zweite Datenblock in den Bereich 960-1022 geschrieben worden ist. Die Daten beider Sprites sind übrigens mit Hilfe des Entwurfsblattes aus Bild 6 berechnet worden.

Das Programm setzt dann die Datenzeiger und VIC.II-Register. Um die Einzelheiten beider Sprites besser erkennbar zu machen, werden beide in doppelter Größe dargestellt. Zeile 1660 setzt die Bits 0 und 1 des Registers VIC+21 auf 1 und macht die beiden Sprites 0 und 1 damit sichtbar. Sollten Sie sich nicht mehr klar darüber sein, warum dazu der Wert 3 benutzt wird, so lesen Sie bitte noch einmal das Kapitel 1.

Die Zeilen 1700-1710 warten auf den üblichen Tastendruck, um das Programm zu beenden, und die Zeilen 1770-1790 setzen die Anzeige- und die Vergrößerungsregister auf den Wert 0 zurück.

Die gleichzeitige Bewegung mehrerer Sprites

Es gibt viele Methoden, um mehrere Sprites zu gleicher Zeit über den Bildschirm zu bewegen. Einige sind einfach zu programmieren, andere schwierig. Einige benötigen viel Speicherplatz, andere wenig. Einige ermöglichen nur geradlinige, andere beliebige Bewegungen. Einige erzeugen schnelle und zusammenhängende, andere nur langsame und ruckartige Bewegungen. Einige sind sehr leicht nachzuvollziehen, andere sind trickreich und schwer verständlich.

Das Beispiel in diesem Kapitel ist ein Mittelding. Es ist nicht zu schwierig, liefert aber trotzdem ein beachtliches Resultat. Um die beiden Sprites zu veranlassen, sich im Kreis herumzujagen, laden Sie bitte zuerst das Programm »Ein Sprite-Paar« und geben dann die in Listing 8 dargestellten Zeilen zusätzlich ein. Speichern und starten Sie dann das neue Programm.

Die beiden Sprites bewegen sich in diesem Programm auf einer quadratischen Bahn in der Bildschirmmitte. Die Seitenlänge dieses Quadrates beträgt 100 Bildpunkte. Seine Eckpunkte sind somit in jeder Richtung 50 Bildpunkte vom Mittelpunkt entfernt. Damit ein Sprite doppelter Größe in der Bildschirmmitte dargestellt werden kann, muß seine horizontale Position 160 und seine vertikale Position 129 betragen. Durch Addition beziehungsweise Subtraktion von 50 wird das Sprite also an einen der Eckpunkte des Bahnquadrates versetzt. Die Positionswerte für die vier Eckpunkte sind in Bild 17 dargestellt.

Sprite 0 beginnt seinen Weg in der linken oberen Ecke und Sprite 1 gegenüber in der rechten unteren Ecke. Sprite 0 bewegt sich nach unten, nach rechts, nach oben und dann nach links. Sprite 1 bewegt sich nach oben, nach links, nach unten und dann nach rechts. Betrachten Sie bitte Bild 18. Es zeigt vier unterschiedliche Bahnpositionen für die beiden Sprites. Bei Bild 1 handelt es sich um die Startposition. Die Pfeile geben die Bewegungsrichtung an. Beachten Sie, daß beide Sprites stets auf gegenüberliegenden Bahnen laufen, jedoch immer in entgegengesetzter Richtung. Diese Bewegungssymmetrie erleichtert die Programmierung erheblich.

Auch bei diesem Beispiel sollten Sie wieder eigene Experimente und Modifikationen am Programm vornehmen und deren Wirkung beobachten.

64er-online.de

```
1000 rem *** ein sprite-jagd ***         <161>
1550 poke vic,110   :rem horzntl.pos. 0  <141>
1560 poke vic+2,210 :rem horzntl.pos. 1  <173>
1570 poke vic+1,79  :rem vertkal.pos. 0  <078>
1580 poke vic+3,179 :rem vertkal.pos. 1  <109>
1662 :                                   <114>
1664 :                                   <116>
1666 rem ** sprite-bewegung vorbereiten  <071>
1667 :                                   <119>
1668 d0 = 1 : d1 = -1                    <240>
1670 :                                   <122>
1672 :                                   <124>
1674 rem ** vertikale bewegung          <058>
1676 :                                   <128>
1678 for move = 1 to 100                 <052>
1680 :   poke vic+1, peek(vic+1) + d0    <205>
1682 :   poke vic+3, peek(vic+3) + d1    <227>
1684 :   get kp$                         <092>
1686 :   if kp$ = "" then 1690           <041>
1688 :     move=100 : keypress = -1      <009>
1690 next move                          <041>
1692 :                                   <144>
1694 :                                   <146>
1696 rem ** bei tastendruck abbrechen    <068>
1698 :                                   <150>
1700 if keypress then 1750               <199>
1702 :                                   <154>
1704 :                                   <156>
1706 rem ** horizontale bewegung         <097>
1708 :                                   <160>
1710 for move = 1 to 100                 <084>
1712 :   poke vic, peek(vic) + d0        <222>
1714 :   poke vic+2, peek(vic+2) + d1    <001>
1716 :   get kp$                         <124>
1718 :   if kp$ = "" then 1722           <248>
1720 :     move=100 : keypress = -1      <041>
1722 next move                          <073>
1724 :                                   <176>
1726 :                                   <178>
1728 rem ** bei tastendruck abbrechen    <100>
1730 :                                   <182>
1732 if keypress then 1750               <231>
1734 :                                   <186>
1736 :                                   <188>
1738 rem ** richtung umgekehrt und los   <031>
1740 :                                   <192>
1742 d0 = -d0 : d1 = -d1                 <222>
1744 goto 1670                           <224>
1746 :                                   <198>
1748 :                                   <200>
```

Listing 8. Änderungen und Ergänzungen zur Umwandlung des Programms »Ein Sprite-Paar« in »Sprite-Jagd«

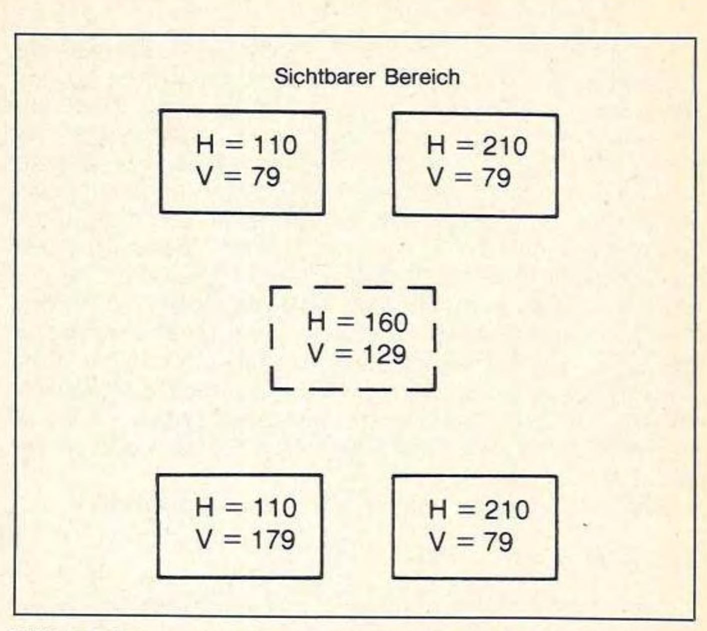


Bild 17. Eckpunkte der quadratischen Bahn der Sprites

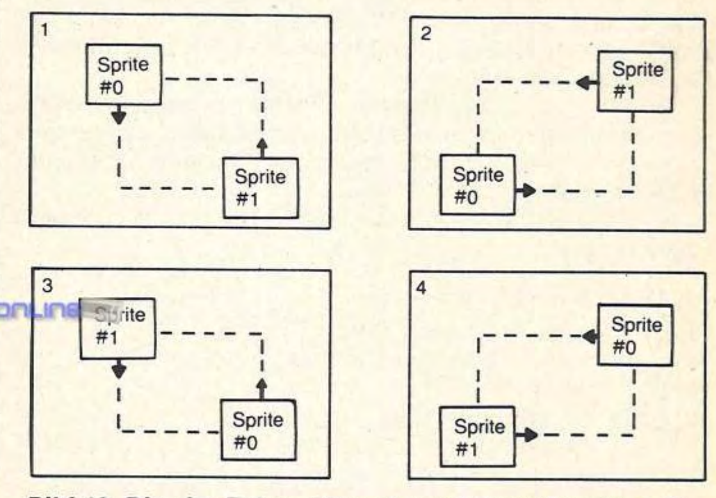


Bild 18. Die vier Eckpositionen der beiden Sprites auf ihrem Bahnquadrat (siehe Programm »Sprite-Jagd«)

Programmierung der Richtungen und Bewegungen

Die Ausgangsstellung der Sprites wird in Zeile 1550-1580 festgelegt:

```
1550 POKE VIC,110   :REM HORIZONTAL 0
1560 POKE VIC+2,210 :REM HORIZONTAL 1
1570 POKE VIC+1,79  :REM VERTIKAL 0
1580 POKE VIC+3,179 :REM VERTIKAL 1
```

Wie bereits beschrieben, startet Sprite 0 in der linken oberen und Sprite 1 in der rechten unteren Ecke.

Das Programm benutzt zur Steuerung der Bewegung zwei Variable, D0 für Sprite 0 und D1 für Sprite 1. Beide werden in Zeile 1668 mit dem Wert 1 initialisiert:

```
1668 D0 = 1: D1 = -1
```

Die Bewegung der Sprites ergibt sich durch Addition der beiden Variablen zu den entsprechenden Positionsregistern. Wenn Sie eine positive Zahl zu dem jeweiligen vertikalen Positionswert addieren, so erhöht sich der Registerwert und das Sprite bewegt sich auf dem Bildschirm nach unten. Bei der Addition negativer Werte ist es umgekehrt, der Registerwert verkleinert sich und das Sprite bewegt sich nach oben. Entsprechend bewirkt die Addition positiver Zahlen zum horizontalen Positionswert eine Bewegung nach rechts, die Addition negativer Werte eine Bewegung nach links.

Zeile 1678-1690 steuert die vertikalen Bewegungen beider Sprites:

```
1678 FOR MOVE = 1 TO 100
1680 :   POKE VIC+1, PEEK(VIC+1) + D0
1682 :   POKE VIC+3, PEEK(VIC+3) + D1
1684 :   GET KP$
1686 :   IF KP$ = "" THEN 1690
1688 :     MOVE = 100 : KEYPRESS = -1
1690 NEXT MOVE
```

Die Zeilen 1678 und 1690 erzeugen eine Schleife, die hundertmal durchlaufen wird. Hundertmal deshalb, weil jede Bahnseite aus 100 Bildpunkten besteht und das Sprite um jeweils einen Bildpunkt weiterbewegt werden soll. Bei jedem Durchlauf wird in Zeile 1680 der Wert 1 der Bewegungsvariablen für Sprite 0 zum vertikalen Positionswert dieses Sprites addiert. In gleicher Weise erfolgt in Zeile 1682 die Addition der Bewegungsvariablen von Sprite 1 zu dessen vertikalem Positionsregister.

Die Zeilen 1686 bis 1688 stellen eine Erweiterung gegenüber den bisherigen Bewegungsprogrammen dar. Die Tastatur wird nun nach jeder Bewegung (um einen Bildpunkt) abgefragt und nicht mehr erst nach Abarbeitung der gesamten Schleife. Die Abfrage erfolgt in Zeile 1686. Wurde keine Taste betätigt, wird Zeile 1688 übersprungen und das Programm fortgesetzt. Ist jedoch eine Taste gedrückt worden, so erhält die Schleifenvariable MOVE den Wert 100 und die Variable KEYPRESS den Wert -1. Die Erhöhung der Schleifenvariablen auf ihren Maximalwert (100) bewirkt, daß die Schleife nicht erneut durchlaufen wird. Dieses ist in Basic der beste Weg, um eine Schleife vorzeitig zu beenden. KEYPRESS wird auf -1 gesetzt, da -1 der Wahrheitswert für eine logisch wahre Bedingung ist. Wir haben dieses Thema in Kapitel 1 behandelt, bitte informieren Sie sich dort, wenn Ihnen etwas unklar ist.

Zeile 1700 steuert den weiteren Programmablauf:

```
1700 IF KEYPRESS THEN 1750
```

Enthält KEYPRESS den Wert 0 (Wahrheitswert für »keine Taste gedrückt«), so beginnt das Programm mit der Schleife für die horizontale Bewegung. Wurde dagegen eine Taste betätigt, so enthält KEYPRESS den Wert -1 und das Programm endet mit dem Programmteil zur Rücksetzung der Register ab Zeile 1750.

Die Abfrage eines Wahrheitswertes bezeichnet man übrigens nach dem englischen Mathematiker Boole als Booleschen Test und die logischen Werte WAHR und FALSCH als Boolesche Variable.

Die Zeilen 1710-1722 bilden die Schleife für die horizontale Bewegung:

```
1710 FOR MOVE = 1 TO 100
1712 :    POKE VIC, PEEK (VIC) + D0
1714 :    POKE VIC+2, PEEK (VIC+2) + D1
1716 :    GET KP$
1718 :    IF KP$ = "" THEN 1722
1720 :       MOVE = 100 : KEYPRESS = -1
1722 NEXT MOVE
```

Diese Schleife entspricht fast vollständig der Schleife für die vertikale Bewegung. Der einzige Unterschied ist, daß die Bewegungsvariablen nicht zu den vertikalen, sondern zu den horizontalen Registern addiert werden.

Zeile 1732 prüft wieder, ob im vorangegangenen Durchgang eine Taste gedrückt worden ist:

```
1732 IF KEYPRESS THEN 1750
```

Wenn ja, endet das Programm mit der Rücksetzung der Register. Anderenfalls beginnt es mit der Änderung der Bewegungsrichtung.

Richtungsänderung beim Bahndurchlauf

Führen Sie sich noch einmal den Weg von Sprite 0 vor Augen. Es muß sich nach unten, nach rechts, nach oben und schließlich nach links bewegen. Anders ausgedrückt: vertikale Bewegung, horizontale Bewegung, vertikale Bewegung, horizontale Bewegung. Die Schleifen für die erste vertikale und horizontale Bewegung haben Sie bereits programmiert. Benötigen Sie nun noch zwei weitere Schleifen?

Nein, Sie können darauf verzichten. Es genügt, die Bewegungsrichtung von Sprite 0 umzukehren und dann die vorhandene horizontale und vertikale Schleife noch einmal zu durchlaufen. Der bisherige Wert der Bewegungsvariablen D0 war 1. Durch Multiplilation mit -1 wird er zu -1. Damit bewegt sich Sprite 0 in der vertikalen Schleife nach oben und in der horizontalen Schleife nach links. Ebenso läßt sich die Bewegungsrichtung von Sprite 1 ändern. Nach einer Multiplikation von D1 mit -1 bewegt es sich in der vertikalen Schleife nach unten und in der horizontalen Schleife nach rechts. Nach der Multiplikation beider Bewegungsvariablen mit -1 muß das Programm also nach Zeile 1678 zurückspringen und die Bewegungsschleifen noch einmal durchlaufen.

Die Zeilen 1742-1744 sorgen für die Bewegungsänderung:

```
1742 D0 = -D0 : D1 = -D1
1744 GOTO 1678
```

Sobald das Programm die Zeile 1742 erneut erreicht, erfolgt wieder ein Richtungswechsel, denn die beiden Richtungswerte werden erneut umgekehrt, das heißt, auf ihren vorangegangenen Wert zurückgesetzt. Da sich beide Sprites in diesem Moment wieder in ihren Ausgangspositionen befinden (Sprite 0 im linken oberen, Sprite 1 im rechten unteren Eckpunkt der Bahn), beginnt die Jagd erneut.

Sie können nun mit dem Programm experimentieren. Wie muß die Änderung für eine Dreiecksbahn aussehen? Lassen sich auch vier Sprites auf der quadratischen Bahn bewegen? Wodurch ergeben sich Spiralbahnen in den Bildmittelpunkt und dann wieder heraus? Bedenken Sie bei größeren Änderungen, daß es wichtig ist, zunächst ein Konzept zu entwickeln und sich dann erst an den Rechner zu setzen.

Zusammenfassung

64ER ONLINE

In diesem Kapitel haben Sie sich mit einigen Techniken zur Steuerung mehrerer Sprites befaßt. Sie haben gelernt:

- unter Benutzung derselben 63 Sprite-Codes mehrere Sprite-Kopien auf dem Bildschirm darzustellen,
- wie mit Hilfe einzelner Bits in einigen VIC.II-Registern bestimmte Sprites gesteuert werden,
- wo die Datenblöcke für mehrere Sprites gespeichert werden können und wie die Datenzeiger in 2040-2047 zu setzen sind,
- auf welche Weise mehrere Sprites gleichzeitig zu bewegen sind,
- wie eine Programmschleife verlassen und der Programmablauf mit Hilfe von Booleschen Variablen gesteuert werden kann.

Im folgenden Kapitel lernen Sie weitere Tricks zur Programmierung von Sprites kennen. Vorher sollten Sie jedoch noch Ihr in diesem Kapitel erworbenes Wissen überprüfen.

Selbsttest

1. Speicherzelle 2045 enthält normalerweise den Datenzeiger für Sprite .....
2. Eine Gruppe von 8 Bit wird ..... genannt.
3. Mit einem Byte können die Dezimalzahlen ..... bis ..... dargestellt werden.
4. Um die Sprites 2, 4 und 7 sichtbar zu machen, müssen Sie den Dezimalwert ..... in das Register VIC+21 schreiben.
5. Um die Sprites 0, 3 und 4 zu verlängern, müssen Sie den Wert ..... in Register VIC+23 schreiben.
6. Die Daten zur Darstellung von acht Sprites werden am besten ab Adresse ..... gespeichert.

64er-online.de

7. Um ein in beide Richtungen vergrößertes Sprite in Höhe der Bildschirmmitte darzustellen, müssen Sie den Wert ..... in das entsprechende vertikale Positionsregister schreiben.
8. Wird der horizontale Positionswert eines Sprites vergrößert, so bewegt sich das Sprite nach .....
9. Variable, deren Zahlenwert den logischen Wert WAHR oder FALSCH repräsentiert, heißen logische oder ..... Variable.
10. Um den Wert einer Variablen abwechselnd auf -1 und 1 zu setzen, muß diese jeweils mit der Zahl ..... multipliziert werden.

Programmierübungen

1. Modifizieren Sie das Programm »Einfache Kopien« derart, daß vier weitere Kopien erscheinen, und zwar in jeder Bildschirmecke eine.
2. Ändern Sie das Programm »Sprite-Jagd«, so daß sich das Sprite-Paar im Uhrzeigersinn bewegt.
3. Ändern Sie das Programm »Sprite-Jagd«, so daß zwei weibliche Sprites zwei männliche Sprites umherjagen.

Antworten zum Selbsttest

Die folgenden Antworten dürften wiederum die zutreffendsten sein. Möglicherweise finden Sie jedoch weitere.

1. 5
2. Byte
3. 255
4. 148
5. 25
6. 12288
7. 129
8. rechts
9. Boolesche
10. -1

Lösungsvorschläge für die Programmierübungen

Die Lösungen bestehen aus zusätzlichen oder geänderten Zeilen für die in diesem Kapitel vorgestellten Programme. Dabei handelt es sich nur um Vorschläge. Anderslautende Lösungen sind ebenfalls richtig, wenn sie den Anforderungen entsprechen.

1. Laden Sie das Programm »Einfache Kopie« und geben Sie folgende Zeilen ein:

```
1000 REM *** ACHT KOPIEN ***
1362 POKE 2044,14     :REM DATENZEIGER 4
1364 POKE 2045,14     :REM DATENZEIGER 5
1366 POKE 2046,14     :REM DATENZEIGER 6
1368 POKE 2047,14     :REM DATENZEIGER 7
1412 POKE VIC+8,24    :REM HORZNTL.POS. 4
1414 POKE VIC+10,64   :REM HORZNTL.POS. 5
1416 POKE VIC+12,24   :REM HORZNTL.POS. 6
1418 POKE VIC+14.64   :REM HORZNTL.POS. 7
1419 POKE VIC+16,160:REM 5&7 MIT 2.HR
1462 POKE VIC+9,50    :REM VERTKAL.POS. 4
1464 POKE VIC+11,50   :REM VERTKAL.POS. 5
1466 POKE VIC+13,299:REM VERTKAL.POS. 6
1468 POKE VIC+15,229:REM VERTKAL.POS. 7
1512 POKE VIC+43,7    :REM 4 IST GELB
1514 POKE VIC+44,5    :REM 5 IST GRUEN
1516 POKE VIC+45,3    :REM 6 IST HELLBLAU
1518 POKE VIC+46,1    :REM 7 IST WEISS
1530 POKE VIC+21,255:REM SPRITES 0-7 AN
```

2. Laden Sie das Programm »Sprite-Jagd« und geben Sie folgende Zeilen ein:

```
1000 REM *** IM UHRZEIGERSINN ***
1674 REM ** HORIZONTALE BEWEGUNG
1680 :   POKE VIC,   PEEK(VIC)   + DO
1682 :   POKE VIC+2, PEEK(VIC+2) + D1
1706 REM ** VERTIKALE BEWEGUNG
1712 :   POKE VIC+1, PEEK(VIC+1) + DO
1714 :   POKE VIC+3, PEEK(VIC+3) + D1
```

3. Laden Sie das Programm »Sprite-Jagd« und geben Sie folgende Zeilen ein:

```
1000 REM *** PAARWEISE JAGD ***
1530 POKE 2041,14    :REM DATENZEIGER 1
1533 POKE 2042,15    :REM DATENZEIGER 2
1536 POKE 2043,15    :REM DATENZEIGER 3
1563 POKE VIC+4,110 :REM HORIZONTAL 2
1566 POKE VIC-6,210 :REM HORIZONTAL 3
1568 :
1583 POKE VIC+5,179 :REM VERTIKAL 2
1586 POKE VIC+7,79  :REM VERTIKAL 3
1613 POKE VIC+41,1  :REM 2 IST WEISS
1616 POKE VIC+42,5  :REM 3 IST GRUEN
1630 POKE VIC+23,15 :REM ALLE 4 SPRITES
1640 POKE VIC+29,15 :REM DOPPELT GROSS
1660 POKE VIC+21,15 :REM ALLE 4 SPR. AN
1674 REM ** BEWEGUNG AUF EINER SEITE
1681 :   POKE VIC+4, PEEK(VIC+4) + DO
1683 :   POKE VIC+6, PEEK(VIC+6) + D1
1706 REM ** BEWEGUNG AUF ANDERER SEITE
1713 :   POKE VIC+5, PEEK(VIC+5) + D1
1715 :   POKE VIC+7, PEEK(VIC+7) + DO
1753 :   POKE VIC+7, PEEK(VIC+7) + DO
```

64ER ONLINE

Kapitel 3: Weitere Sprite-Tricks

Interessiert es Sie, wie mehrfarbige Sprites erzeugt werden? In diesem Kapitel werden Sie Näheres darüber erfahren. Außerdem lernen Sie, Sprites übereinander hinweg zu bewegen. Schließlich werden Sie sogar, mit Hilfe mehrerer sich überlagernder Grundfiguren, eine kleine Zirkusnummer programmieren. Nebenbei gibt es weitere Experimente mit Bits, Bytes und Sprites.

Ein normales Sprite ist durch 63 Bytes definiert, die ab einer bestimmten Adresse im Speicher abgelegt sind. Jedes Byte besteht aus acht Bits, und jedes Bit steuert einen Bildpunkt (sichtbar oder unsichtbar). Ein Sprite besteht somit aus (63 * 8) = 504 Bildpunkten.

Hat das einem bestimmten Bildpunkt zugeordnete Bit den Wert 1, so wird der Bildpunkt in der im Farbregister des Sprites codierten Farbe ausgegeben. Ist der Wert 0, so wird der Bildpunkt in der Farbe des Bildschirmes angezeigt, das heißt, er ist unsichtbar.

Im Normalfall gibt es für einen Bildpunkt also nur zwei Zustände: sichtbar oder unsichtbar. Der C64 bietet jedoch noch eine weitere Möglichkeit, den sogenannten Mehrfarbenmodus. In diesem Modus werden zwei Bits (ein Bitpaar) benutzt, um eine Farbe zu bestimmen.

Zwei Bits lassen sich in vier unterschiedlichen Bitmustern darstellen (Bild 19). Damit können vier Farben für die beiden dem Bitpaar zugeordneten Bildpunkte definiert werden.

64er-online.de 64er-online.net

Selbstverständlich sind beide Bildpunkte von gleicher Farbe. Am einfachsten ist es, die beiden Bildpunkte als einen großen Bildpunkt doppelter Breite zu betrachten. Bei der Benutzung dieser Möglichkeit bestimmt dann ein Byte die Farbe für vier Doppelbildpunkte, und eine Sprite-Zeile besteht statt aus 24 Bildpunkten nun aus 12 Doppelbildpunkten. Die horizontale Auflösung eines mehrfarbigen Sprites ist also nur halb so hoch wie die eines normalen Sprites.

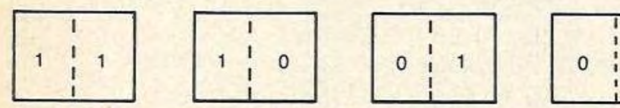


Bild 19. Mit zwei Bit können vier unterschiedliche Bitmuster dargestellt werden

Halbe Auflösung bedeutet, daß nicht mehr jeder einzelne Bildpunkt sichtbar oder unsichtbar gemacht werden kann, sondern nur noch Bildpunktpaare. Zur Wiederholung: Der Grund dafür ist, daß zwei Bits benötigt werden, um eine Farbe zu definieren, und ein Byte somit nur noch vier Doppelbildpunkte steuern kann (Bild 20). Eine Sprite-Reihe besteht jetzt also aus 12 Doppelbildpunkten. Die Gesamtgröße des Sprites wird dadurch nicht verändert. Ein mehrfarbiges Sprite besteht somit aus (63 * 4) 252 Doppelbildpunkten und wird nach wie vor durch 63 Datenbytes definiert.

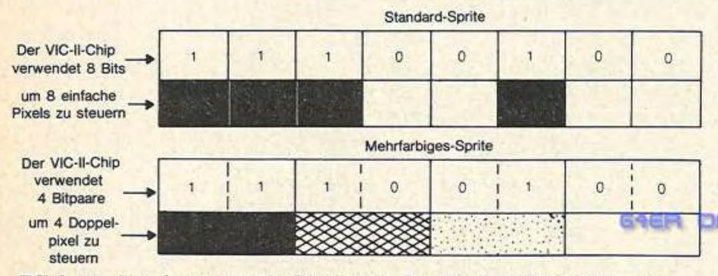


Bild 20. In einem mehrfarbigen Sprite wird jeweils ein Doppelpunkt durch ein Bitpaar gesteuert

Der Mehrfarbenmodus

Welche Farben stehen für mehrfarbige Sprites zur Verfügung? Das Bitpaar 00 definiert die Farbe des Bildschirmes, die ja bekanntlich im Register VIC+33 (53281) codiert ist. Das Bitpaar 01 bedeutet, daß der Farbcode aus dem Mehrfarbenregister 0 mit der Adresse VIC+37 gelesen wird. Ein Bitpaar 10 veranlaßt den C64, das normale Farbregister des Sprites zu benutzen. Wie Sie sich erinnern, besitzt jedes Sprite sein eigenes Farbregister im Bereich VIC+39 bis VIC + 46. Das Bitpaar 11 schließlich verweist auf den Farbcode im Mehrfarbregister 1 mit der Adresse VIC + 38. Alle Zuordnungen sind in Tabelle 6 noch einmal zusammengefaßt.

Auf welche Weise wird dem C64 nun mitgeteilt, daß er ein bestimmtes Sprite mehrfarbig darstellen soll? Das Steuerregister dafür ist VIC+28. Wie üblich, ist jedem Bit ein Sprite zugeordnet, Bit 0 steuert Sprite 1, und so weiter. Ist ein Bit dieses Registers auf 1 gesetzt, so wird das entsprechende Sprite mehrfarbig dargestellt. Ein Rücksetzen des Bits bewirkt wieder die normale Darstellung des Sprites.

Bitpaar	Beschreibung	Adresse
0 0	Bildschirmfarbe	VC+33 (53281)
0 1	Sprite-Mehrfarbenregister 0	VC+37 (53285
1 0	Sprite-Farbregister	eines der Register VC+39 - VC+46 (53287 - 53294)
1 1	Sprite-Mehrfarbenregister 1	VC+38 (53286)

Tabelle 6. Das einem Doppelbildpunkt zugeordnete Bitpaar bestimmt das Farbcode-Register

Der Entwurf eines mehrfarbigen Sprites

Falls Sie nicht zu den wenigen Spezialisten gehören, die ein mehrfarbiges Sprite im Kopf entwerfen können, benutzen Sie bitte das Entwurfsblatt aus Bild 21. Es entspricht weitgehend dem Entwurfsblatt für einfarbige Sprites aus Kapitel 1. Wie dieses enthält es 21 Zeilen, die Bitwerte und drei Spalten für die Codezahlen. Allerdings sind nur 12 Spalten vorhanden, da in mehrfarbigen Sprites ja nur Doppelbildpunkte angesprochen werden können.

Spaltennummer	0	1	2	3	4	5	6	7	8	9	10	11	Zahlencodes
Wert	128 64	32 16	8 4	2 1	128 64	32 16	8 4	2 1	128 64	32 16	8 4	2 1	
Zeile 0													
Zeile 1													
Zeile 2													
Zeile 3													
Zeile 4													
Zeile 5													
Zeile 6													
Zeile 7													
Zeile 8													
Zeile 9													
Zeile 10													
Zeile 11													
Zeile 12													
Zeile 13													
Zeile 14													
Zeile 15													
Zeile 16													
Zeile 17													
Zeile 18													
Zeile 19													
Zeile 20													

Hintergrundfarbe (transparent) 0 0 — Mehrfarben-register 0 0 1 — Sprite-Farbe 1 0 — Mehrfarben-register 1 1 1

Bild 21. Ein Entwurfsblatt für mehrfarbige Sprites

Wie wird dieses Entwurfsblatt benutzt? Betrachten Sie dazu Bild 22, in dem ein ausgefülltes Entwurfsblatt dargestellt ist.

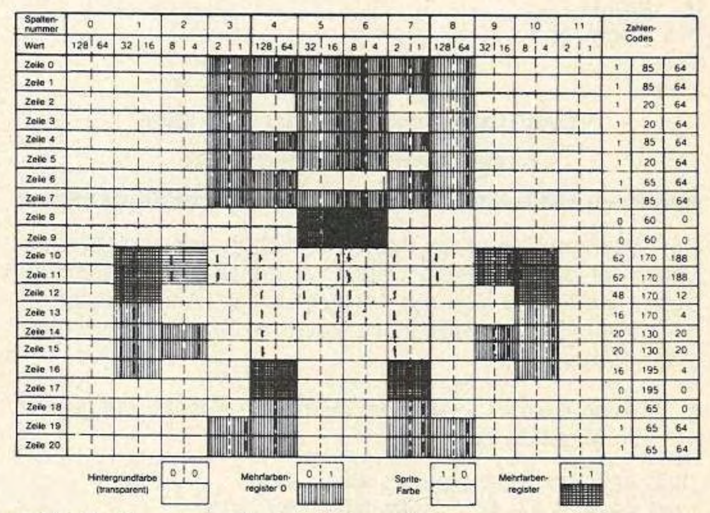


Bild 22. Ein ausgefülltes Entwurfsblatt

Denken Sie sich zunächst für jede Farbe, die Sie benutzen wollen, ein Muster aus, und tragen Sie dieses in die Farbkästchen am unteren Rand des Blattes ein. Sie können auch Farbstifte verwenden. Die Farbe des Bildschirmes wird am besten durch ein weißes Kästchen dargestellt.

Nun setzen Sie das Sprite aus Doppelkästchen zusammen, wobei ein Doppelkästchen einem Doppelbildpunkt entspricht. Liegt die Form des Sprites fest, füllen Sie die Doppelkästchen mit dem entsprechenden Farbmuster, am besten eine Farbe nach der anderen. Anschließend beginnen Sie dann mit der Berechnung der Zahlencodes.

Dazu addieren Sie die in der zweiten Kopfzeile des Entwurfsblattes dargestellten Bitcodes aller mit einem Farbmu-

ster ausgefüllten Einzelkästchen (!) jeder Gruppe (jeder Zahlencode gilt ja für eine Gruppe von 8 Bildpunkten bzw. 4 Doppelbildpunkten), wenn das entsprechende Bit in den Farbkästchen den Wert 1 hat. Diese Addition entspricht der Addition bei einfarbigen Sprites, nur werden diesmal Doppelkästchen betrachtet, deren Einzelkästchen bei der Berechnung unberücksichtigt bleiben, wenn über dem entsprechenden Einzelkästchen des Farbdoppelkästchens der Bitwert 0 steht.

Die errechneten Zahlencodes tragen Sie wie üblich in die am rechten Rand des Entwurfsblattes dafür vorgesehenen Spalten ein.

Fertigen Sie sich mehrere Kopien des Entwurfsblattes an, und zeichnen Sie eigene Entwürfe. Vergewissern Sie sich, daß Sie die Berechnung der Zahlencodes in Bild 22 verstanden haben, und berechnen Sie dann die Codes für Ihren eigenen Entwurf. Sie benötigen diese Daten im folgenden Kapitel.

Ein Programm zur Darstellung mehrfarbiger Sprites

Listing 9 zeigt das Programm »Vierfarben-Sprite«. Es erzeugt das in Bild 22 gezeichnete Sprite. Die Betätigung einer beliebigen Taste beendet das Programm.

Das Programm entspricht weitgehend den bisherigen Sprite-Programmen, mit Ausnahme der Zeilen 1400-1440:

```
1400 POKE VIC+28,1  :REM MEHRFARBMODUS
1410 POKE VIC+33,0  :REM HGRUND SCHWARZ
1420 POKE VIC+37,7  :REM MFREG 0 GELB
1430 POKE VIC+39,5  :REM SPRITE 0 GRUEN
1440 POKE VIC+38,6  :REM MFREG 1 BLAU
```

Zeile 1400 schreibt in das Auswahlregister den Wert 1, so daß Sprite 0 mehrfarbig dargestellt wird. Die Zeilen 1410 bis 1440 bestimmen dann die zu benutzenden Farben: Schwarz, definiert durch das Bitpaar 00; Gelb, definiert durch das Bitpaar 01; Grün, definiert durch das Bitpaar 10; und Blau, definiert durch das Bitpaar 11.

Am Ende des Programms wird das Mehrfarben-Sprite-Register MFSR zurückgesetzt:

```
1580 POKE VIC+28,0  :REM MEHRFARB. AUS
```

Die Benutzung eines normalen Sprites als mehrfarbiges Sprite ergibt eine sehr eigenartige Darstellung. Sie können sich davon überzeugen, indem Sie eines der vorangegangenen Sprite-Programme laden und die Zeilen 1410-1440 hinzufügen.

Geben Sie das Programm »Vierfarben-Sprite« ein, speichern und starten Sie es anschließend. Versuchen Sie, die Farbwerte in den Zeilen 1410-1440 zu modifizieren und andere Kombinationen zu finden.

Nachdem Sie diese Experimente beendet haben, ist es Zeit, Ihre eigenen Daten zu testen. Ersetzen Sie die Bildpunktdaten in den Zeilen 1150-1250 des Programms »Vierfarben-Sprite« durch die von Ihnen im letzten Abschnitt errechneten Werte, und starten Sie das Programm anschließend erneut. Entspricht das Ergebnis Ihren Erwartungen? Manchmal muß man etwas herumbasteln, ehe die Daten mit dem gewünschten Bild übereinstimmen.

Sprites unterschiedlicher Prioritäten

Wenn sich Sprites über den Bildschirm bewegen, kann es vorkommen, daß sie miteinander kollidieren und sich überschneiden. In diesem Fall bestimmt der C 64, in welcher

```
1000 rem *** vierfarben-sprite ***                <054>
1010 :                                             <224>
1020 :                                             <234>
1030 rem ** ausgabe programmeldung                <013>
1040 :                                             <000>
1050 print "{clr,8down}bitte warten"              <034>
1060 :                                             <020>
1070 :                                             <030>
1080 rem ** sprite-daten laden                    <212>
1090 :                                             <050>
1100 for n = 896 to 958                           <018>
1110 :   read spdta                               <061>
1120 :   poke n, spdta                            <098>
1130 next n                                       <238>
1140 :                                             <100>
1150 data   1, 85, 64,  1, 85, 64                 <234>
1160 data   1, 20, 64,  1, 20, 64                 <240>
1170 data   1, 85, 64,  1, 20, 64                 <252>
1180 data   1, 65, 64,  1, 85, 64                 <199>
1190 data   0, 60,  0,  0, 60,  0                 <028>
1200 data  62,170,188, 62,170,188                 <190>
1210 data  48,170, 12, 16,170,  4                 <203>
1220 data  20,130, 20, 20,130, 20                 <180>
1230 data  16,195,  4,  0,195,  0                 <184>
1240 data   0, 65,  0,  1, 65, 64                 <106>
1250 data   1, 65, 64                             <010>
1260 :                                             <220>
1270 :                                             <230>
1280 rem ** sprite-register vorbereiten           <000>
1290 print "{clr}"       :rem schirm loesch
                                                   <069>
1310 poke 2040,14   :rem datenzeiger              <243>
1320 vic = 53248    :rem grafikbaustein           <135>
1330 :                                             <036>
1340 poke vic,160   :rem horiz.position           <007>
1350 poke vic+1,129 :rem vertk.positian           <227>
1360 :                                             <066>
1370 poke vic+23,1  :rem verlaengerung            <026>
1380 poke vic+29,1  :rem verbreiterung            <235>
1390 :                                             <096>
1400 poke vic+28,1  :rem mehrfarbmodus            <190>
1410 poke vic+33,0  :rem hgrund schwarz           <070>
1420 poke vic+37,7  :rem mfreg 0 gelb             <182>
1430 poke vic+39,5  :rem sprite 0 gruen           <078>
1440 poke vic+38,6  :rem mfreg 1 blau             <119>
1450 :                                             <156>
1460 poke vic+21,1  :rem sprite 0 ein             <225>
1470 :                                             <176>
1480 :                                             <186>
1490 rem ** prog.ende mit tastendruck             <255>
1500 :                                             <206>
1510 get kp$                   0                  <089>
1520 if kp$ = "" then 1510                        <191>
1530 :                                             <236>
1540 :                                             <248>
1550 rem ** ruecksetzen des registers             <137>
1560 :                                             <012>
1570 poke vic+21,0  :rem sprite   aus             <133>
1580 poke vic+28,0  :rem mehrfarb.aus             <233>
1590 poke vic+29,0  :rem verbreit.aus             <011>
1600 :                                             <052>
1610 :                                             <062>
1620 end                                          <098>
```

Listing 9. Ausdruck des Programms »Vierfarben-Sprite«

Rangfolge die Sprites dargestellt werden, das heißt, welches der beiden Sprites verdeckt und welches angezeigt wird. Die höchste Priorität hat immer Sprite 0, die niedrigste Sprite 7. Bei einer Überlagerung von Sprite 0 und Sprite 7 wird also Sprite 0 dargestellt. Ebenso hat zum Beispiel Sprite 4 eine höhere Priorität als Sprite 5. Bild 23 zeigt die Rangfolge der Sprites.

Bei der vollständigen oder teilweisen Überlagerung zweier Sprites werden alle Bildpunkte des Sprites mit der höheren Priorität, die unsichtbar, also in der Farbe des Bildschirmes dargestellt werden, sozusagen transparent, so daß die entsprechenden Teile des darunterliegenden Sprites wie durch ein Fenster sichtbar sind.

Listing 10 zeigt das Programm »Sprite-Überlagerung«. Geben Sie es ein, speichern und starten Sie es.

»Sprite Überlagerung« stellt vier gleichartige Sprites auf dem Bildschirm dar und verschiebt sie dann so lange überein-

64er-online.de

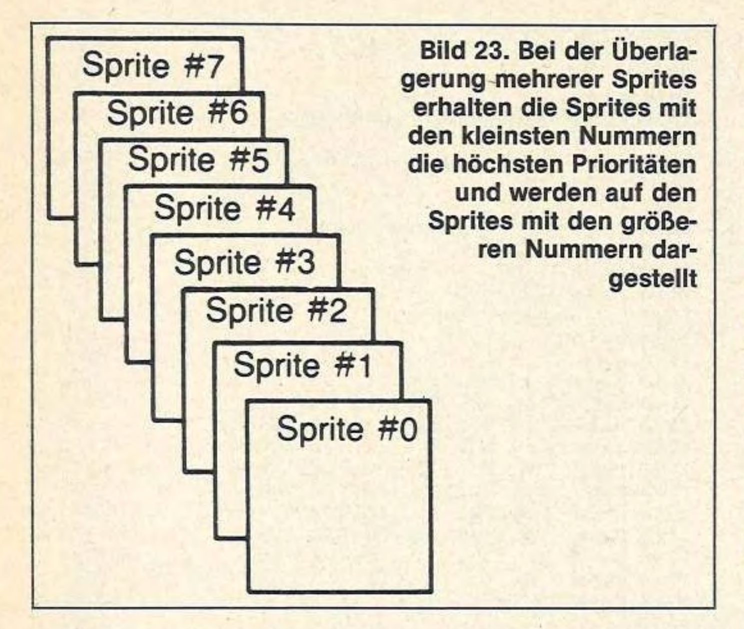


Bild 23. Bei der Überlagerung mehrerer Sprites erhalten die Sprites mit den kleinsten Nummern die höchsten Prioritäten und werden auf den Sprites mit den größeren Nummern dargestellt

ander, bis eine Taste gedrückt wird. Beachten Sie, wie bei der Überlagerung Teile der unteren Sprites durch die transparenten Bereiche der Sprites 0 und 1 hindurch sichtbar werden.

Abgesehen von der Überlagerung von Sprites ist in dem Programm auch interessant, auf welche Weise die Gestalt der Sprites definiert und der Bewegungsablauf programmiert wird.

Die Zahlencodes für das Sprite im Programm »Sprite-Überlagerung« werden durch eine Programmschleife generiert:

```
1100 FOR N = 832 TO 894
1110 :    POKE N, 60
1120 NEXT N
```

Eine ähnliche Technik haben Sie bereits im Programm »ein einfaches Sprite« benutzt. Dort enthielt der POKE-Befehl den Wert 255, und jeder Bildpunkt des Sprites wurde sichtbar. Im vorliegenden Fall bewirkt die Zahl 60, daß nur die mittleren vier von jeweils 8 Bildpunkten sichtbar werden. Dieser Effekt ist in Bild 24 dargestellt. Enthält jede Sprite-Zeile drei Muster dieser Art, so besteht das Sprite aus drei vertikal verlaufenden Streifen.

Mit Hilfe einer Programmschleife lassen sich viele Sprites erzeugen. Testen Sie zum Beispiel folgende Zeilen:

```
1100 FOR N = 832 TO 894 STEP 2
1110 :    POKE N, 255 : POKE N+1, 0
```

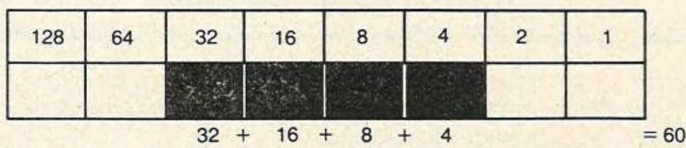


Bild 24. Der Zahlencode 60 für ein Sprite bewirkt, daß die vier mittleren der acht Bildpunkte jeder Gruppe sichtbar werden

```
1000 rem *** sprite-ueberlagerung ***        <162>
1010 :                                       <224>
1020 :                                       <234>
1030 rem ** ausgabe programmeldung           <013>
1040 :                                       <000>
1050 print "{clr,8down}bitte warten"         <034>
1060 :                                       <020>
1070 :                                       <030>
1080 rem ** sprite-daten laden               <212>
1090 :                                       <050>
1100 for n = 832 to 894                      <124>
1110 :    poke n, 60                         <016>
1120 next n                                  <228>
1130 :                                       <090>
1140 :                                       <100>
1150 rem ** sprite-register vorbereiten      <124>
1160 :                                       <120>
1170 print "{clr}"        :rem schirm loesch
     .                                       <203>
1180 vic = 53248      :rem grafikbaustein    <249>
1190 :                                       <150>
1200 poke 2040,13    :rem datenzeiger 1      <025>
1210 poke 2041,13    :rem datenzeiger 2      <228>
1220 poke 2042,13    :rem datenzeiger 3      <174>
1230 poke 2043,13    :rem datenzeiger 4      <121>
1240 :                                       <200>
1250 poke vic,226    :rem horzntl.pos. 0     <235>
1260 poke vic+2,94   :rem horzntl.pos. 1     <160>
1270 poke vic+4,144 :rem horzntl.pos. 2      <182>
1280 poke vic+6,176 :rem horzntl.pos. 3      <227>
1290 :                                       <252>
1300 poke vic+1,140 :rem vertal.pos. 0       <109>
1310 poke vic+3,118 :rem vertal.pos. 1       <174>
1320 poke vic+5,190 :rem vertal.pos. 2       <155>
1330 poke vic+7,68  :rem vertal.pos. 3       <150>
1340 :                                       <046>
1350 poke vic+39,7  :rem 0 ist gelb          <083>
1360 poke vic+40,5  :rem 1 ist gruen         <098>
1370 poke vic+41,3  :rem 2 ist hellblau      <002>
1380 poke vic+42,1  :rem 3 ist weiss         <182>
1390 :                                       <096>
1400 poke vic+23,15 :rem alle sprites        <080>
1410 poke vic+29,15 :rem dopp. groesse       <108>
1420 :                                       <126>
1430 poke vic+21,15 :rem sprite 0-3 an       <182>
1440 :                                       <146>
1450 :                                       <156>
1460 rem ** beweg.register vorbereiten       <144>
1470 rem    und anf.schritte zuweisen        <208>
1480 :                                       <186>
1490 mr(0) = vic   : mr(1) = vic+2           <154>
1500 mr(2) = vic+5 : mr(3) = vic+7           <041>
1510 :                                       <216>
1520 mv(0) = -1 : mv(1) = 1                  <201>
1530 mv(2) = -1 : mv(3) = 1                  <147>
1540 :                                       <248>
1550 df = -1 :rem -1=n.innen 0=n.aussen      <205>
1560 :                                       <012>
1570 :                 0                     <022>
1580 rem ** sprite bewegen                   <059>
1590 :                                       <042>
1600 for count = 1 to 200                    <007>
1610 :    sprnum = int((count-1)/50)         <150>
1620 :    if df then sprnum = 3 - sprnum     <144>
1630 :    reg = mr(sprnum)                   <245>
1640 :    move = mv(sprnum)                  <140>
1650 :    poke reg, peek(reg) + move         <084>
1660 :    get kp$                            <068>
1670 :    if kp$= "" then 1690               <025>
1680 :       count = 200 : keypress = -1     <009>
1690 next count                              <037>
1700 :                                       <152>
1710 :                                       <162>
1720 rem ** prog.end nach tastendruck        <246>
1730 :                                       <182>
1740 if keypress then 1900                   <239>
1750 :                                       <202>
1760 :                                       <212>
1770 rem ** pause, dann richtung um-         <069>
1780 rem    kehren und von vorn              <182>
1790 :                                       <242>
1800 for delay = 1 to 400 : next delay       <212>
1810 for sprnum = 0 to 3                     <022>
1820 :    mv(sprnum) = -1 * mv(sprnum)       <102>
1830 next sprnum                             <177>
1840 df = -1 - df                            <018>
1850 goto 1600                               <108>
1860 :                                       <058>
1870 :                                       <068>
1880 rem ** register zuruecksetzen           <115>
1890 :                                       <088>
1900 poke vic+21,0                           <146>
1910 poke vic+29,0                           <164>
1920 poke vic+23,0                           <168>
1930 :                                       <128>
1940 end                                     <164>
```

Listing 10. Ausdruck des Programms »Sprite-Überlagerung«

Sicherlich können Sie mit Hilfe dieser Technik noch eine Vielzahl weiterer Sprites erfinden. Starten Sie die neue Version Ihres Programms, und lassen Sie sich nicht hypnotisieren. Versuchen Sie, herauszufinden, wie viele weitere Sprites Sie durch den geschickten Einsatz von Programmschleifen erzeugen können.

Der Bewegungsablauf

Beim Start des Programmes »Sprite-Überlagerung« befinden sich die vier Sprites in den in Bild 25 dargestellten Grundpositionen. Nacheinander bewegen sie sich dann auf die Bildmitte zu, legen dort eine Pause ein, wandern anschließend auf ihre Ausgangspositionen zurück und beginnen den Reigen nach einer weiteren Pause erneut.

Für die Programmierung von Bewegungsabläufen ist es von Vorteil, die auszuführende Bewegung auf gleichartige Abläufe hin zu untersuchen. Im vorliegenden Fall muß jedes Sprite nur zwei Bewegungen ausführen: sich in den Bildmittelpunkt bewegen und auf gleicher Bahn zurückkehren. Daher kann ein Programmsegment, das diese Bewegungen steuert, von allen vier Sprites benutzt werden. Sind diese Bewegungsvariablen als Feldvariable definiert, so sind die Bewegungen besonders einfach zu programmieren: für das entsprechende Sprite muß als Index nur die entsprechende Sprite-Nummer angegeben werden.

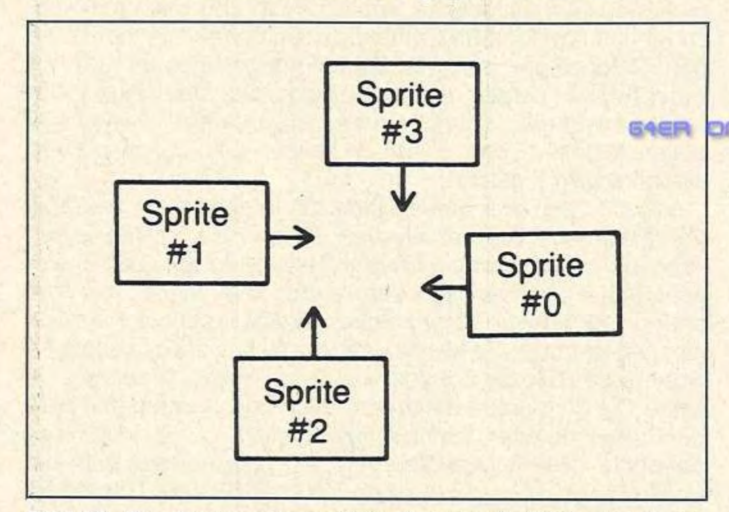


Bild 25. Die Ausgangsposition der vier Sprites aus dem Programm »Sprite-Überlagerung«. Diese Pfeile geben die Bewegungsrichtung nach dem Programmstart an.

Eine weitere Vereinfachung ergibt sich aus der Tatsache, daß die beiden Bewegungen, die ein Sprite auszuführen hat, sich nur in der Richtung unterscheiden. Nach Abschluß einer Bewegung muß also nur der Wert der Bewegungsvariablen umgekehrt werden. Damit ist dann im Prinzip die Steuerung aller vier Sprites durch ein Programmsegment möglich. Nur die Feinheiten müssen noch genauer ausgearbeitet werden (ein berühmtes letztes Wort vieler Programmierer).

Da ein Sprite sich entweder nur horizontal oder nur vertikal bewegt, ist ein Positionsregister pro Sprite ausreichend. In den Zeilen 1490-1500 werden die entsprechenden Registeradressen an vier Variable übergeben:

```
1490 MR(0) = VIC   :MR(1) = VIC+2
1500 MR(2) = VIC+5 :MR(3) = VIC+7
```

Betrachten Sie noch einmal Bild 25. Sprite 0 und 1 bewegen sich horizontal, Sprite 2 und 3 vertikal. Entsprechend diesen Richtungen wurden die Positionsregister in den Zeilen 1490-1500 ausgewählt.

In den Zeilen 1520-1530 werden die Bewegungsvariablen definiert:

```
1520 MV(0) = -1  : MV(1) = 1
1530 MV(2) = -1  : MV(3) = 1
```

Der jeweilige Wert der Richtungsvariablen, addiert zum aktuellen Positionswert, ergibt die neue Position des entsprechenden Sprites. Betrachten Sie dazu noch einmal das Bild 25. Die Pfeile kennzeichnen die Richtung der Sprites zu Beginn des Programms. Sprite 3 zum Beispiel beginnt mit einer Bewegung nach unten. Nach jedem Schritt erhöht sich sein Positionswert, und genau diese Vergrößerung des Wertes wird in Zeile 1530 vorgenommen. Um die Bewegungsrichtung umzukehren, genügt es, den Wert der Variablen MV(3) mit –1 zu multiplizieren. Dann nämlich verkleinert sich der vertikale Positionswert bei jedem Schritt um 1, und Sprite 3 bewegt sich wieder hinauf.

Es ist noch eine weitere Einzelheit zu berücksichtigen: die Reihenfolge der Sprites. Bei der Bewegung nach innen (auf den Mittelpunkt zu) soll die Bewegung in der Reihenfolge 3, 2, 1, 0 erfolgen, bei der Bewegung nach außen (vom Mittelpunkt weg) dagegen in der Reihenfolge 0, 1, 2, 3. Zeile 1550 initialisiert eine Variable DF, mit der die Reihenfolge gesteuert wird:

```
1550 DF = -1 :REM -1=N.INNEN 0=N.AUSSEN
```

Die allgemeine Bewegungsschleife

Die Zeilen 1600-1690 bilden die Bewegungsschleife:

```
1600 FOR COUNT = 1 TO 200
1610 :   SPRNUM = INT((COUNT-1)/50)
1620 :   IF DF THEN SPRNUM = 3 -SPRNUM
1630 :   REG = MR(SPRNUM)
1640 :   MOVE = MV(SPRNUM)
1650 :   POKE REG, PEEK(REG) + MOVE
1660 :   GET KP$
1670 :   IF KP$ = "" THEN 1690
1680 :      COUNT = 200 : KEYPRESS = -1
1690 NEXT COUNT
```

Die Zeilen 1600 und 1690 definieren eine Schleife, die 200mal durchlaufen wird, falls Sie nicht zwischenzeitlich eine beliebige Taste betätigen. Die Schleifenvariable hat den Endwert 200, da jedes der vier Sprites einen Weg von 50 Bildpunkten zurückzulegen hat (4 * 50 = 200).

Die Zeilen 1610 und 1620 bestimmen, welches Sprite bewegt werden soll, und speichern die entsprechende Sprite-Nummer in der Variablen SPRNUM. Bei der Bewegung nach innen hat DF den Wert –1, und SPRNUM nimmt nacheinander die Werte 3, 2, 1 und 0 an. Bei der Bewegung nach außen, wenn DF den Wert 0 hat, erhält SPRNUM die Werte 0, 1, 2 und 3.

Zeile 1630 wählt, entsprechend dem Wert von SPRNUM, das Positionsregister, und Zeile 1640 die Schrittweite. In Zeile 1650 wird dann die Schrittweite zum alten Positionswert addiert.

Zeile 1660 überprüft die Tastatur. Wurde eine Taste betätigt, beendet Zeile 1670 die Schleife. Sie haben diese Technik schon im vorhergehenden Abschnitt kennengelernt. Nachdem die Sprites ihre Bewegung nach innen oder außen beendet haben, muß die Bewegungsrichtung umgekehrt werden:

```
1000 rem *** jongleur ***                        <060>
1010 :                                           <224>
1020 :                                           <234>
1030 rem ** ausgabe programmeldung               <013>
1040 :                                           <000>
1050 print "{clr,8down}bitte warten";            <152>
1060 :                                           <020>
1070 :                                           <030>
1080 rem ** sprite-daten laden                   <212>
1090 :                                           <050>
1100 for n = 832 to 1023                         <109>
1110 :    read spdta                             <061>
1120 :    if spdta = -1 then print " ."; :
     goto 1140                                   <121>
1130 :    poke n, spdta                          <108>
1140 next n                                      <248>
1150 :                                           <110>
1160 data   0, 16,  0,  0,  0,  0                <231>
1170 data   1,  0,128,  0,  0,  0                <090>
1180 data   0,  0,  0,  0,120,  0                <047>
1190 data   4,120, 16,  0,120,  0                <229>
1200 data   0,120,  0, 12, 24,  0                <114>
1210 data  15,255, 16,  0, 61,128                <168>
1220 data   0, 60,176,  4, 24,240                <044>
1230 data   0, 61,  0,  0, 60,  0                <132>
1240 data   0, 36,  0,  0, 36,  0                <255>
1250 data   0, 36,  0,  0, 36,  0                <009>
1260 data   0,102,  0, -1                        <072>
1270 :                                           <230>
1280 data   0,  8,  0,  0, 64,  0                <070>
1290 data   0,  0,  0,  0,  0, 64                <224>
1300 data   0,  0,  0,  4, 60,  0                <033>
1310 data   0, 60,  0,  0, 60,  0                <150>
1320 data   0, 60, 16,  8, 24,  0                <212>
1330 data  13,255,  0, 15, 61, 48                <088>
1340 data   0, 61,240,  0, 24,  0                <052>
1350 data   0,190,  0,  0, 60,  0                <196>
1360 data   0, 36,  0,  0, 36,  0                <121>
1370 data   0, 36,  0,  0, 36,  0                <131>
1380 data   0,102,  0, -1                        <194>
1390 :                                           <096>
1400 data   0, 32,  0,  0,  2,  0                <056>
1410 data   0,  0,  0,  2,  0,  0                <143>
1420 data   0,  0, 32,  0, 60,  0                <200>
1430 data   0, 60,  0,  0, 60,  0                <014>
1440 data   8, 60,  0,  0, 24, 16                <078>
1450 data   0,255,152,  1,188,248                <227>
1460 data  13, 60,  0, 15, 24, 64                <093>
1470 data   0, 60,  0,  0, 60,  0                <054>
1480 data   0, 36,  0,  0, 36,  0                <241>
1490 data   0, 36,  0,  0, 38,  0                <060>
1500 data   0, 96,  0, -1                        <198>
1510 :                                           <216>
1520 :                                           <226>
1530 rem ** sprite register vorbereiten          <180>
1540 :                                           <248>
1550 print "{clr}"       :rem schirm loesch
     .                                           <075>
1560 vic = 53248      :rem grafikbaustein        <121>
1570 :                                           <022>
1580 poke 2040,13     :rem datenzeiger 0         <025>
1590 poke vic,160     :rem horzntl.pos. 0        <186>
1600 poke vic+1,129   :rem vertkal.pos. 0        <104>
1610 poke vic+39,1    :rem 0 ist weiss           <196>
1620 poke vic+29,1    :rem sprite 0 hat          <060>
1630 poke vic+23,1    :rem dopp. groesse         <141>
1640 poke vic+21,1    :rem sprite 0 ein          <151>
1650 :                                           <102>
1660 :                                           <112>
1670 rem ** jonglieren beginnt                   <141>
1680 :                                           <132>
1690 image = peek (2040) +1                      <079>
1700 if image = 16 then image = 13               <247>
1710 poke 2040, image                            <202>
1720 :                                           <172>
1730 for delay = 1 to 30 : next delay            <198>
1740 :                                           <192>
1750 :                                           <202>
1760 rem ** prog.ende nach tastendruck           <027>
1770 :                                           <222>
1780 get kp$                                     <105>
1790 if kp$ = "" then 1690                       <087>
1810 poke vic+21,0    :rem sprites aus           <218>
1820 poke vic+29,0    :rem und wieder            <048>
1830 poke vic+23,0    :rem normalgroesse         <251>
1840 :                                           <038>
1850 end                                         <074>
```

Listing 11. Ausdruck des Programms »Jongleur«

```
1810 FOR SPRNUM = 0 TO 3
1820 :    MV(SPRNUM) = -1 * MV(SPNUM)
1830 NEXT SPRNUM
1840 DF = -1 - DF
1850 GOTO 1600
```

Zunächst wird die Bewegungsrichtung jedes Sprites durch Multiplikation mit -1 umgekehrt, dann, in Zeile 1840, der Variablenwert für die Reihenfolge angepaßt. 0 wird zu -1, und -1 zu 0. Zeile 1850 schließlich übergibt die Steuerung wieder an die Bewegungsschleife, die in Zeile 1600 beginnt. Auf diese Weise bewegen sich die Sprites vor und zurück, bis ein Tastendruck ihren Tanz unterbricht.

Die vorliegende Bewegungsschleife bietet eine Vielzahl von Variationsmöglichkeiten. Versuchen Sie, andere Bewegungsformen zu realisieren. Dazu einige Vorschläge:

* Bewegen Sie zwei Sprites gleichzeitig.
* Bewegen Sie die Sprites bei der Bewegung nach außen auf neue Ausgangspositionen zu.
* Überlagern Sie die Sprites in der Bildmitte vollständig.

Trickbilder

In allen bisherigen Programmen wurde die Bewegung der Sprites durch eine Veränderung des jeweiligen Positionswertes bewirkt. Das Sprite selbst änderte dabei seine Form nicht. Diesen Effekt können Sie jedoch hervorrufen, wenn Sie mehrere unterschiedliche Sprites laden und dann nacheinander an derselben Bildschirmposition darstellen, indem Sie den Datenzeiger in einer Schleife nacheinander auf die Datenblöcke zeigen lassen. Erfolgt die Darstellung der einzelnen Sprites schnell genug, so entsteht, ähnlich wie beim Trickfilm, der Eindruck einer ständig die Form wechselnden Figur.

Bild 26 zeigt drei Einzelbilder, die in schnellem Wechsel übereinandergeblendet werden sollen. Sie stellen einen Jongleur in drei verschiedenen Stellungen dar. Die Bewegungsfolge beginnt nach dem dritten Bild wieder mit dem ersten. Der Entwurf einer solchen Bildfolge gelingt meistens nicht auf Anhieb. Entwerfen Sie zunächst die Einzelbilder, wobei jedes Bild die Fortsetzung des vorangegangenen sein sollte. Die Bildpunktdaten berechnen Sie wie immer mit Hilfe der Entwurfsblätter. Sie benötigen diese Daten im folgenden Abschnitt. Hier einige Themen: ein springender Ball, ein zwinkerndes Auge, ein lachendes Gesicht, ein blinkender Stern, eine sich verlängernde und verkürzende Linie, ein Blizzard.

Listing 11 enthält das Programm »Jongleur«. Es stellt die in Bild 26 gezeigten Bilder in der beschriebenen Weise auf dem Bildschirm dar.

Die Zeilen 1100-1130 generieren die Sprite-Daten. Zeile 1120 enthält eine interessante Einzelheit:

```
1120 :    IF SPDTA = -1 THEN PRINT "."; : GOTO 1140
```

Ein Sprite wird durch 63 Datenbytes definiert. Wie inKapitel 2 beschrieben, zeigt der Datenzeiger jedoch immer auf einen Block von jeweils 64 Bytes. Liegen die Datenblöcke direkt hintereinander, so lassen sich die Daten mit einer einzigen Schleife nacheinander laden, wenn jeder Datenblock mit einem vierundsechzigsten Füllbyte aufgefüllt wird. Geben Sie nun diesem Füllbyte einen Wert, der in einem Datenblock normalerweise nicht vorkommt, so kann das Programm das Ende jedes Datenblocks erkennen und eine entsprechende Meldung ausgeben. Im vorliegenden Programm hat das Füllbyte den Wert -1.

Die drei Datenblöcke der Sprite-Bilder haben die Adressen 832-894, 896-958 und 960-1022. Durch Division der An-

64er-online.de

fangsadresse jedes Blocks durch 64 ergeben sich die Werte 13, 14 und 15 für die Datenzeiger. Das Programm führt die Überlagerung der Bilder aus, indem es dem Datenzeiger für Sprite 0 (Adresse 2040) in einer Schleife nacheinander die Werte 13, 14 und 15 zuordnet und diese Schleife ständig wiederholt.

In den Zeilen 1580-1640 werden die Sprite-Register wie gewohnt initialisiert. Da der Datenzeiger für Sprite 0 hier den Wert 13 erhält, beginnt die Vorstellung mit dem im Datenblock 832-894 gespeicherten Bild.

Die Zeilen 1690-1710 tauschen die Bilder aus:

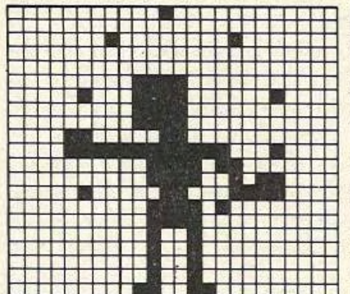
```
1690 IMAGE = PEEK (2040) +1
1700 IF IMAGE = 16 THEN IMAGE =13
1710 POKE 2040, IMAGE
```

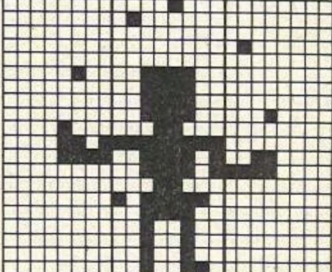
In Zeile 1690 wird der Wert des aktuellen Datenzeigers um 1 erhöht. Hat er den Wert 16 erreicht, setzt Zeile 1700 ihn auf 13 zurück. In Zeile 1710 erfolgt die Eintragung des Wertes in die entsprechende Speicherzelle. Der Zeiger durchläuft somit die Werte 13, 14, 15 und beginnt dann erneut mit 13.

Zeile 1730 ist nur eine Verzögerungsschleife. Durch Veränderung des Schleifenendwertes wird der Jongleur schneller oder langsamer.

Die Zeilen 1780-1790 enthalten schließlich die gewohnte Tastaturabfrage. Wurde keine Taste betätigt, so wird die Steuerung an Zeile 1690 übergeben und das nächste Bild angezeigt. Anderenfalls endet das Programm nach Rücksetzung der Datenzeiger.

Ersetzen Sie im Programm »Jongleur« nun die Daten in den Zeilen 1160-1500 durch die Codes für Ihre eigenen Trickbilder, und starten Sie das Programm. Stimmt der Ablauf mit Ihren Vorstellungen überein? Experimentieren Sie mit der Reihenfolge der Einzeldarstellungen, ändern Sie den Endwert der Verzögerungsschleife, benutzen Sie andere Daten. Durch derartige Experimente lernen Sie die Anwendung der beschriebenen Programmtechnik.





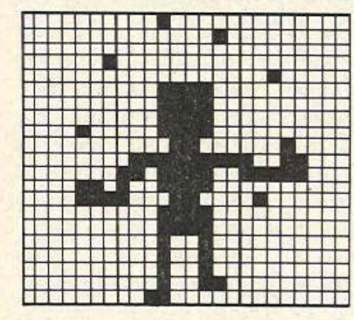


Bild 26. Anfangsbild nach dem Start des Programms

```
1000 rem *** lustige kollision ***          <044>
1010 :                                      <224>
1020 :                                      <234>
1030 rem ** sprite-daten laden              <162>
1040 :                                      <000>
1050 print "{clr}"                          <022>
1060 for n = 832 to 958                     <102>
1070 :   read spdta                         <021>
1080 :   poke n, spdta                      <058>
1090 next n                                 <198>
1100 :                                      <060>
1110 data   0,255,  0,  1,129,128           <200>
1120 data   3,  0,192,  3,  0,192           <104>
1130 data   3,  0,192,  6,102, 96           <005>
1140 data  60,102, 60, 96,  0,  6           <246>
1150 data 192,  0,  3,192,102,  3           <061>
1160 data 198, 60, 99, 99,  0,198           <001>
1170 data 113,129,142, 28,195, 56           <052>
1180 data  12,195, 48, 12,102, 48           <119>
1190 data   6, 60, 96,  6,  0, 96           <194>
1200 data   3,129,192,  0,195,  0           <177>
1210 data   0,126,  0,  0                   <157>
1220 :                                      <180>
1230 data   0,  0,  0,  0, 16,  0           <009>
1240 data   0, 56,  0,  0, 84,  0           <079>
1250 data   0, 16,  0,  2, 16,128           <211>
1260 data   1, 17,  0,  0,146,  0           <107>
1270 data  16, 84, 16, 32, 56,  8           <248>
1280 data 127,255,252, 32, 56,  8           <195>
1290 data  16, 84, 16,  0,146,  0           <190>
1300 data   1, 17,  0,  2, 16,128           <079>
1310 data   0, 16,  0,  0, 84,  0           <023>
1320 data   0, 56,  0,  0, 16,  0           <113>
1330 data   0,  0,  0                       <182>
1340 :                                      <046>
1350 :                                      <056>
1360 rem ** sprites vorbereiten und ein     <096>
1370 :                                      <076>
1380 vic = 53248     :rem grafikbaustein    <195>
1390 poke vic+33,0   :rem hgrnd schwarz     <092>
1400 :                                      <106>
1410 poke 2040,13    :rem datenzeiger 0     <109>
1420 poke 2041,14    :rem datenzeiger 1     <057>
1430 :                                      <136>
1440 poke vic,120    :rem horzntl.pos. 0    <030>
1450 poke vic+2,160  :rem horzntl.pos. 1    <079>
1460 poke vic+1,138  :rem vertkal.pos. 0    <214>
1470 poke vic+3,126  :rem vertkal.pos. 1    <209>
1480 :                                      <186>
1490 poke vic+39,3  :rem 0 ist hellblau     <193>
1500 poke vic+40,7  :rem 1 ist gelb         <129>
1510 poke vic+29,2  :rem nur 1 ist von      <187>
1520 poke vic+23,2  :rem dopp. groesse      <033>
1530 :                                      <236>
1540 poke vic+21,3  :rem beide sichtb.      <064>
1550 :                                      <002>
1560 :                                      <012>
1570 rem ** bewegung sprite 0               <223>
1580 :                                      <032>
1590 jr = peek (56320)  :rem eingang 2      <203>
1600 if (jr and 16) = 0 then 1870           <120>
1610 hd = sgn(jr and 4) - sgn(jr and 8)     <172>
1620 vd = sgn(jr and 1) - sgn(jr and 2)     <202>
1630 :                                      <082>
1640 poke vic, peek(vic) + hd               <026>
1650 poke vic+1, peek(vic+1) + vd           <006>
1660 :                                      <112>
1670 :                                      <122>
1680 rem ** zurueck, wenn keine kollis.     <205>
1690 :                                      <142>
1700 if peek(vic+30) = 0 then 1490          <100>
1710 :                                      <162>
1720 :                                      <172>
1730 rem ** kollision : 1 wird weiss
        und 0 blinkt bunt                   <249>
1740 :                                      <192>
1750 poke vic+40, 1                         <254>
1760 :                                      <212>
1770 hue = peek(vic+39) and 15              <143>
1780 hue = hue + 1                          <143>
1790 if hue = 8 then hue = 1                <070>
1800 poke vic+39, hue                       <021>
1810 :                                      <008>
1820 goto 1590                              <095>
1830 :                                      <028>
1840 :                                      <038>
1850 rem ** ausgangswert und ende           <157>
1860 :                                      <058>
1870 poke vic+21,0                          <116>
1880 poke vic+29,0                          <134>
1890 poke vic+23,0                          <138>
1900 :                                      <098>
1910 end                                    <134>
```

Listing 12. Das Programm »Lustige Kollision«

64er-online.de

Sprite-Kollisionen

Es ist für viele Anwendungen nützlich zu wissen, wann bestimmte Objekte auf dem Bildschirm miteinander kollidieren. Bei den früheren Heimcomputern war dies sehr schwierig festzustellen. Der Commodore 64 enthält jedoch Schaltkreise, die in der Lage sind, solche Kollisionen zu erkennen.

Die Kollision zweier Sprites wird in Register VIC+30 (Adresse 53278) registriert. Jedes Bit dieses Register ist einem Sprite zugeordnet. Ist ein Sprite in eine Kollision mit einem anderen Sprite verwickelt, erhält sein Bit den Wert 1. Kollidiert zum Beispiel Sprite 2 mit Sprite 7, so werden die Bits 2 und 7 des Registers VIC+30 auf 1 gesetzt. Eine Rücksetzung dieser Bits auf 0 erfolgt nicht dadurch, daß beide Sprites sich wieder voneinander entfernen, sondern nur durch einen PEEK-Befehl, mit dem das Register VIC+30 gelesen wird.

Das Register VIC+31 (Adresse 53279) registriert Kollisionen zwischen Sprites und Hintergrundobjekten. Hintergrundobjekte sind entweder Textzeichen (Textmodus) oder Teile einer hochauflösenden Grafik (Grafikmodus). Wiederum ist jedem Sprite ein Bit des Registers zugeordnet. Das Bit wird bei einer Kollision auf 1 gesetzt. Kollidiert zum Beispiel Sprite 5 im Textmodus mit einem Textzeichen, so erhält Bit 5 in Adresse VIC+31 den Wert 1. Das Bit behält diesen Wert, bis das Register mit einem PEEK-Befehl gelesen wird.

Listing 12 enthält das Programm »Lustige Kollision«. Es enthält Beispiele zum Lesen der Steuerknüppelregister und zur Registrierung der Kollisionen zweier Sprites. Geben Sie das Programm ein, speichern und starten Sie es. Daraufhin werden die in Bild 27 dargestellten Sprites sichtbar. Benutzen Sie einen an Eingang 2 angeschlossenen Steuerknüppel, um das Gesicht auf der Wetterfahne zu bewegen. Beachten Sie, was geschieht, wenn die beiden Sprites kollidieren. Ein Druck auf den Feuerknopf beendet das Programm.

Die Zeilen 1050-1090 laden die Daten für beide Sprites, die Zeilen 1380-1540 initialisieren die erforderlichen VIC-Register und aktivieren die Sprites.

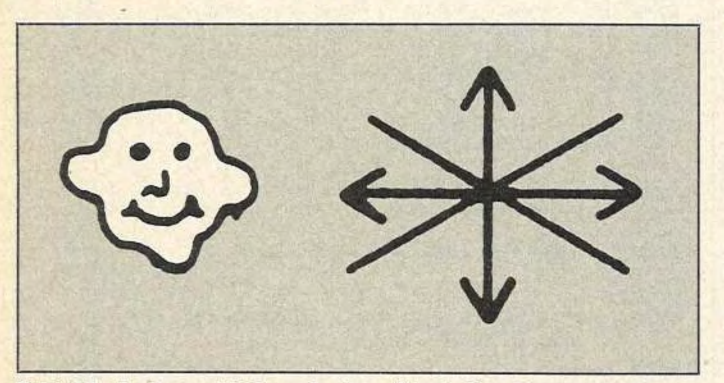


Bild 27. Anfangsbild nach dem Start des Programms »Lustige Kollision«

Nun folgt der Hauptteil des Programms. Zeile 1590 liest den Wert des E/A-Registers mit der Adresse 56320. Wie Sie vielleicht wissen, ist diese Adresse dem Steuerknüppel in Eingang 2 zugeordnet. In Zeile 1600 wird eine AND-Funktion benutzt, um festzustellen, ob der Feuerknopf gedrückt ist. Ist das der Fall, so endet das Programm mit der Rücksetzungsroutine ab Zeile 1870.

Die Zeilen 1610-1620 bestimmen aus dem Wert in Register 56320 die auszuführende horizontale und vertikale Bewegung, und zwar auf eine schnelle und trickreiche Weise. Mit Hilfe der AND-Funktion ergeben sich die Werte der den einzelnen Schaltern zugeordneten Bits. Die SGN-Funktion liefert entweder den Wert 0 oder den Wert 1, je nachdem, ob der Wert in Klammern gleich oder größer als 0 ist. Je nach Stellung des Steuerknüppels erhält HD, die Variable für die horizontale Bewegung, den Wert -1, 0 oder 1. Die gleichen Werte kann VD, die Variable für die vertikale Bewegung, annehmen. Beide Bewegungsvariablen werden dann benutzt, um die neue Position für Sprite 0 festzulegen.

Zeile 1700 überprüft das Register, in dem die Kollision zweier Sprites registriert wird. Findet keine Kollision statt, springt das Programm zurück, aktiviert wieder die alten Sprite-Farben und fragt erneut das Steuerknüppelregister ab. Ist dagegen eine Kollision erfolgt, so werden in den Zeilen 1750-1800 erst die Sprite-Farben geändert, ehe das Programm zur Abfrage des Steuerknüppels springt.

Zusammenfassung

In diesem Kapitel haben Sie gelernt:

- die VIC.II-Register so zu programmieren, daß ein Sprite in vier Farben dargestellt wird,
- ein mehrfarbiges Sprite zu entwerfen,
- zu erkennen, welche Sprites bei einer Überlagerung sichtbar bleiben,
- weitere Einzelheiten, um mehrere Sprites gleichzeitig zu bewegen,
- mit Hilfe eines durch eine Programmschleife gesteuerten Datenzeigers eine Tricksequenz aus mehreren Sprites zu erzeugen.
- Sprite-Kollisionen abzufragen

Ein Kurs dieses Umfangs kann Ihnen nur Anfangskenntnisse in der Programmierung von Sprites vermitteln. Durch Experimente mit den vorgestellten und mit eigenen Programmen werden Sie Ihre Kenntnisse jedoch schnell vertiefen.

Selbsttest

1. Im Mehrfarbenmodus wird ein Doppelbildpunkt durch zwei Bits definiert. Damit lassen sich ..... Farben darstellen.
2. Da Sprites im Mehrfarbenmodus nur eine Breite von 12 Doppelbildpunkten besitzen, ist ihre ..... Auflösung kleiner als die eines einfarbigen Sprites.
3. Welche Sprites werden durch den Befehl POKE VIC+28,15 im Mehrfarbenmodus dargestellt ?
4. Welches Sprite hat bei der Überschneidung mehrerer Sprites die höchste Priorität ?
5. Welches Aussehen haben die Sprites, die sich im Programm »Sprite-Überlagerung« durch folgende Programmzeilen ergeben:

```
     1100 FOR N = 832 TO 894 STEP 3
1105 :    POKE N, 225
1110 :    POKE N+1, 195
1115 :    POKE N+2, 135
```

6. Betrachten Sie die Zeilen 1610-1620 im Programm »Sprite-Überlagerung«. Welchen Wert erhält SPRNUM in den Zeilen 1610 und 1620, wenn COUNT den Wert 120 und DF den Wert 0 hat ?
7. Wie viele Punkte (.) werden neben der Programmeldung im Programm »Jongleur« auf dem Bildschirm ausgegeben, während die Sprite-Daten gelesen werden?

8. Welche Wirkung hat eine Änderung des Endwertes der Verzögerungsschleife von 30 auf 100 im Programm »Jongleur«?

Programmierübungen

1. Modifizieren Sie das Programm »Vierfarben-Sprite« derart, daß aus den vorhandenen Sprite-Codes ein zweites Sprite erzeugt und ebenfalls im Mehrfarbenmodus angezeigt wird.
2. Modifizieren Sie das Programm »Sprite-Ueberlagerung« derart, daß sich die vier Sprites in der Bildschirmmitte vollständig überlagern.
3. Ändern Sie das Programm »Jongleur«, so daß sich die Kugeln des Jongleurs zunächst im Uhrzeigersinn bewegen, nach einer Weile die Richtung ändern, sich dann wieder rechts herum bewegen, und so weiter.

Antworten zum Selbsttest

1. vier
2. horizontale
3. 0, 1, 2 und 3
4. 0
5. Jedes Sprite besteht aus vier vertikalen Streifen (Bild 28).
6. 2
7. 3
8. Die Bewegungen verlangsamen sich.

Lösungsvorschläge für die Programmierübungen

Die folgenden Lösungen sind Änderungen oder Erweiterungen der in diesem Kapitel besprochenen Programme.

1. Laden Sie das Programm »Vierfarben-Sprite«, und geben Sie die folgenden Zeilen ein:

Bild 28. Die in Aufgabe 5 des Selbsttests genannten Programmzeilen ergeben dieses Sprite

```
1000 REM *** ZWEI VIERFARBEN-SPRITES ***
1315 POKE 2041,14      :REM ZEIGER SPRITE1
1353 POKE VIC+2,160    :REM SPRITE 1 HPOS.
1356 POKE VIC+3,69     :REM SPRITE 1 VPOS.
1370 POKE VIC+23,3     :REM VERLAENGERN
1380 POKE VIC+29,3     :REM VERBREITERN
1400 POKE VIC+28,3     :REM 0+1 MEHRFARBEN
1435 POKE VIC+40,2     :REM SPRITE 1 ROT
1460 POKE VIC+21,3     :REM SPRITE 0+1 EIN
```

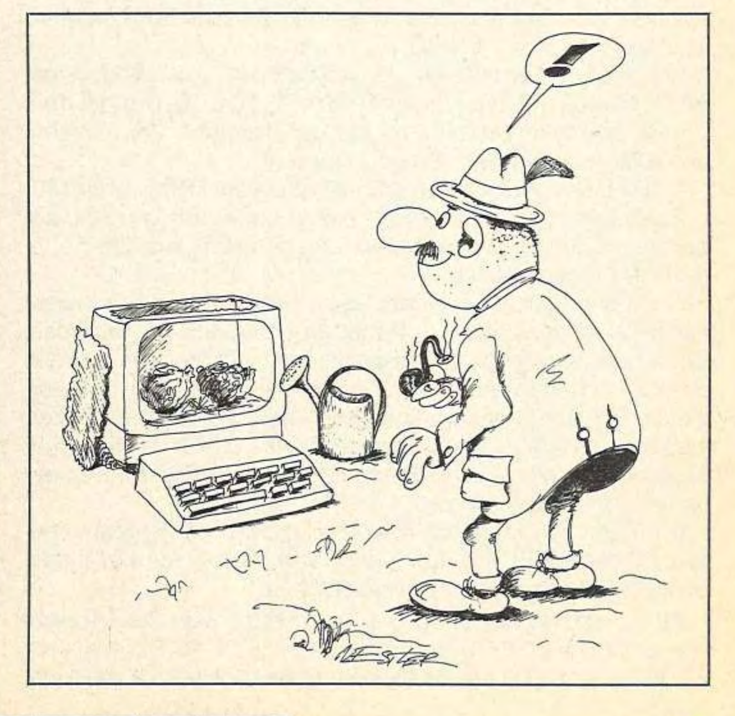
2. Laden Sie das Programm »Sprite-Überlagerung« und geben Sie die folgenden Zeilen ein:

```
1000 REM *** TOTALUEBERLAGERUNG ***
1250 POKE VIC,210     :REM HORZNTL.POS. 0
1260 POKE VIC+2,110 :REM HORZNTL.POS. 1
1270 POKE VIC+4,160 :REM HORZNTL.POS. 2
1280 POKE VIC+6,160 :REM HORZNTL.POS. 3
1300 POKE VIC+1,129 :REM VERTKAL.POS. 0
1310 POKE VIC+3,129 :REM VERTKAL.POS. 1
1320 POKE VIC+5,179 :REM VERTKAL.POS. 2
1330 POKE VIC+7,79  :REM VERTKAL.POS. 3
```

3. Laden Sie das Programm »Jongleur«, und geben Sie die folgenden Zeilen ein:

```
1000 REM *** RECHTS-LINKS-JONGLEUR ***
1655 JUGDIR = 1       :REM IM UHRZEIGERS.
1690 IMAGE = PEEK (2040) + JUGDIR
1705 IF IMAGE = 12 THEN IMAGE = 15
1712 :
1715 COUNT = COUNT + 1
1718 IF INT (COUNT/27) = COUNT/27
     THEN JUGDIR = -JUGDIR : COUNT = 0
```

Damit sind wir am Ende unseres Sprite-Kurses angelangt. Was Ihnen jetzt noch zu tun bleibt, ist das hier Gelernte in praktische Programme umzusetzen, etwa in Spiele. Natürlich können dabei noch Schwierigkeiten auftreten, aber mit diesem Kurs haben Sie alles nötige Wissen über Sprites an der Hand. (Stan Krute/ev)



64ER ONLINE

Wie wär's mit:

Iδ ιόχειρη – Γραφή

Hätten Sie gewußt, daß die Hieroglyphen in der Überschrift »Zeichen selber machen« bedeuten? Wir zeigen Ihnen, wie Sie solche oder ähnliche Zeichen selbst gestalten können und was dabei zu berücksichtigen ist.

Es gibt viele Anwendungen, die das Arbeiten mit einem abgeänderten Zeichensatz erforderlich machen. Vielleicht wollen Sie mathematische Sonderzeichen darstellen, ein griechisches Textverarbeitungs-Programm schreiben oder Spiele programmieren, die einen abgeänderten Zeichensatz benutzen. Wie auch immer, zuerst müssen wir uns die Frage stellen: »Was ist bei der Gestaltung eines Zeichensatzes zu beachten?«

Wir müssen uns überlegen, wo der neue Zeichensatz liegen soll. Möchte man kein vom Basic aus nutzbares RAM verwenden, bleibt nur der Bereich von $A000 bis $FFFF. Aber genau das ist für viele Programmierer die Schwierigkeit. Der Grund dafür ist der, daß der im C 64 eingebaute Videocontroller immer nur eine 16-KByte-Bank »sieht«. Das heißt, daß der Bildschirm, also das Video-RAM und der Zeichensatz in der gleichen 16-KByte-Bank liegen müssen. Eine Ausnahme bildet dabei die Einschaltkonfiguration. Denn hier liegt der Bildschirm bekanntlich bei $400 und der Zeichensatz bei $D000 unter dem I/O-Bereich, in dem die Register der Peripheriebausteine liegen, zum Beispiel auch die des Videocontrollers (Bild 1). Die Einschaltkonfiguration ist aber eine hardwaremäßige Maßnahme und läßt sich von einem Programm aus nicht nachbilden. Der Videocontroller beziehungsweise das Zeichensatz-ROM ist nämlich so verdrahtet, daß der Zeichensatz vom Videocontroller aus gesehen bei $1000 liegt. Vom Prozessor aus gesehen liegt er aber bei $D000. Das ist übrigens auch der Grund dafür, warum sich ein beliebiger Zeichensatz nicht ins RAM nach $1000 legen läßt. Denn die Register im Controller sind so gesetzt, als läge der Zeichensatz tatsächlich ab $1000.

Wir wollen den neuen Zeichensatz in das RAM nach $E000 legen und das Video-RAM nach $CC00. Das hat den Vorteil, daß kein nutzbares RAM verlorengeht. Wir müssen uns aber jetzt um zwei Dinge kümmern:

1. Wie bekomme ich den Zeichensatz vom ROM ins RAM?
2. Welche Speicherzellen müssen verändert werden, so daß der Computer den neuen Zeichensatz und das neue Video-RAM akzeptiert?

Es ist unmöglich, mit einem Basic-Programm den Originalzeichensatz auszulesen. Denn das Basic-ROM und das Betriebssystem müssen ausgeblendet werden, um auf die Bank zu schalten, in der der Zeichensatz liegt (Bild 1). Dazu existiert im Computer die Speicherzelle 1. Wir wollen uns hier kurz mit den Bits 0, 1 und 2 beschäftigen, die für das Ein- und Ausblenden der verschiedenen RAM- oder ROM-Konfigurationen verantwortlich sind.

Bit 0: Ist dieses Bit auf »0« gesetzt, wird der Bereich von $A000 bis $BFFF auf RAM geschaltet. Daraus folgt, daß das Basic-ROM ausgeblendet wird (Bild 2).

Bit 1: Wird dieses Bit auf »0« gesetzt, so wird der Bereich von $A000 bis $BFFF und zusätzlich noch der Bereich von $E000 bis $FFFF auf RAM und der Bereich von $D000 bis $DFFF auf Zeichensatz-ROM geschaltet. Dies ist die Speicherbelegung, in der wir den Zeichensatz auslesen können. Bei dieser Konfiguration existiert aber weder das Basic noch das Betriebssystem (Bild 3).

Bit 2: Dieses Bit blendet, wenn es auf »0« gesetzt ist, das gesamte ROM aus. Hier existieren also volle 64 KByte RAM (Bild 4).

Die Routine zum Kopieren des Zeichensatzes läßt sich, wie oben erwähnt, nicht in Basic schreiben. Es bleibt also nur ein Maschinenprogramm. Um aber auch dem Anfänger das Programm verständlich zu machen, möchte ich zuerst ein Basic-Programm vorstellen, mit dem das Kopieren des Zeichensatzes denkbar wäre. Anschließend wird dieses Programm in Maschinensprache umgesetzt.

Also zuerst das Basic-Programm:

```
10 POKE1,PEEK(1)AND253:REM BIT 1 AUSBLENDEN
20 VON=53248:NACH=57344
30 FOR X=0 TO 4096: REM ANZAHL DER ZU KOPIERENDEN
BYTES
40 POKE NACH+X,PEEK(VON+X)
50 NEXT
60 POKE1,PEEK(1)OR2:REM BIT 1 EINBLENDEN
```

Nun das entsprechende Assembler-Programm:

```
10   -.EQ VON=53248
20   -.EQ NACH=57344
30   -.EQ X=4096
40   -.BA $C000
50   -      LDA #<(VON)  ;»VON« WIRD IM HI/LO-BYTE
                          FORMAT
60   -      LDX #>(VON)
70   -      STA $FA      ;NACH $FA/$FB GESPEICHERT
80   -      STX $FB
90   -      LDA #<(NACH);DAS GLEICHE FUER »NACH«
100  -      LDX #>(NACH)
110  -      STA $FC
120  -      STX $FD
130  -      LDY #0       ;ZAEHLER MIT NULL
                          VORBELEGEN
140  -      STY $FE
150  -      STY $FF
151  -      LDA 1        ;BIT 2 AUSBLENDEN
152  -      AND #253
153  -      STA 1
160  -LOOP  LDA ($FA),Y  ;DEN AKKU MIT »VON« LADEN
170  -      STA ($FC),Y  ;UND NACH »NACH« SPEICHERN
180  -      INC $FA      ;»VON« UM 1 ERHOEHEN
190  -      BNE WE
200  -      INC $FB
210  -WE    INC $FE      ;»NACH« UM 1 ERHOEHEN
220  -      BNE WE1
230  -      INC $FF
240  -WE1   INC $FE      ;ZAEHLER UM 1 ERHOEHEN
250  -      BNE WE2
260  -      INC $FF
270  -WE2   LDA $FE      ;ZAEHLER-»X«
280  -      SEC
290  -      SBC #<(X)
300  -      LDA $FF
310  -      SBC #>(X)
320  -      BCC LOOP     ;IST DAS ERGEBNIS NEGATIV
                          DANN NACH LOOP
330  -      LDA 1        ;ANSONSTEN BIT 2 WIEDER
                          EINBLENDEN
340  -      ORA #2
350  -      STA 1
360  -      RTS          ;UND ZURUECK INS BASIC
```

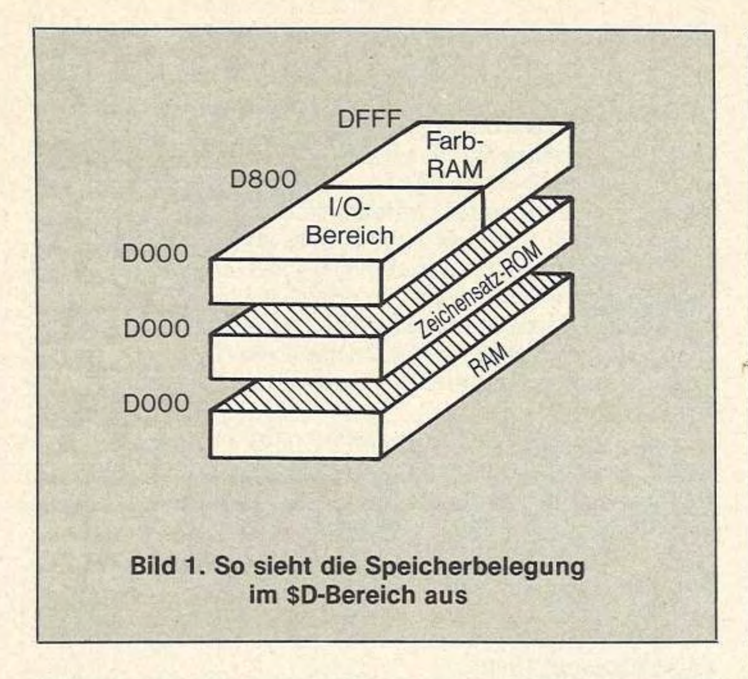


Bild 1. So sieht die Speicherbelegung im $D-Bereich aus

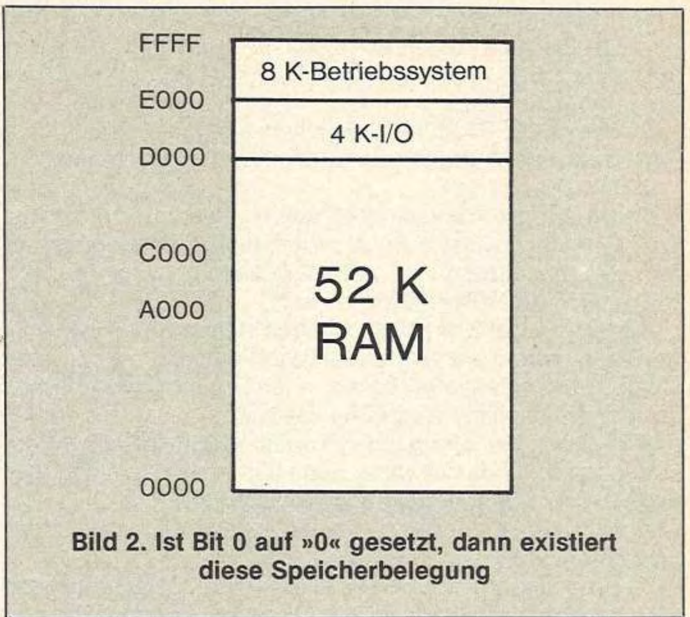


Bild 2. Ist Bit 0 auf »0« gesetzt, dann existiert diese Speicherbelegung

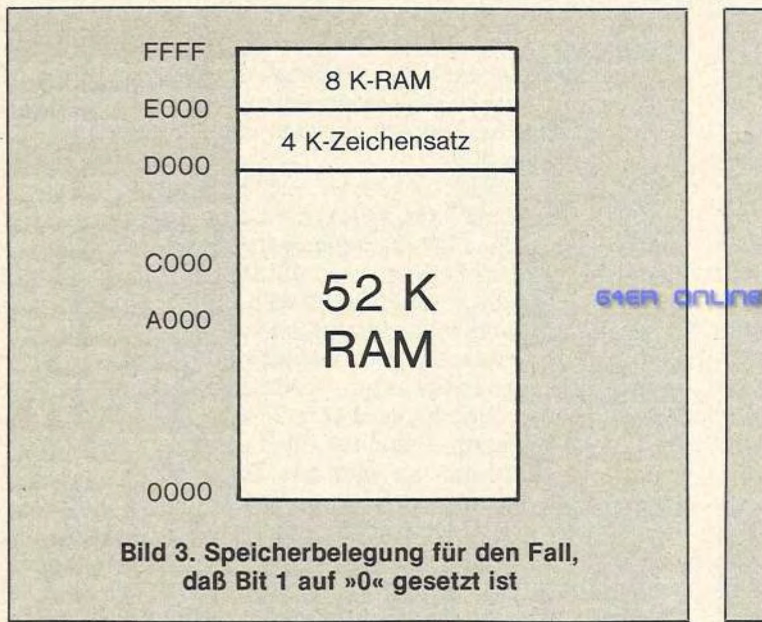


Bild 3. Speicherbelegung für den Fall, daß Bit 1 auf »0« gesetzt ist

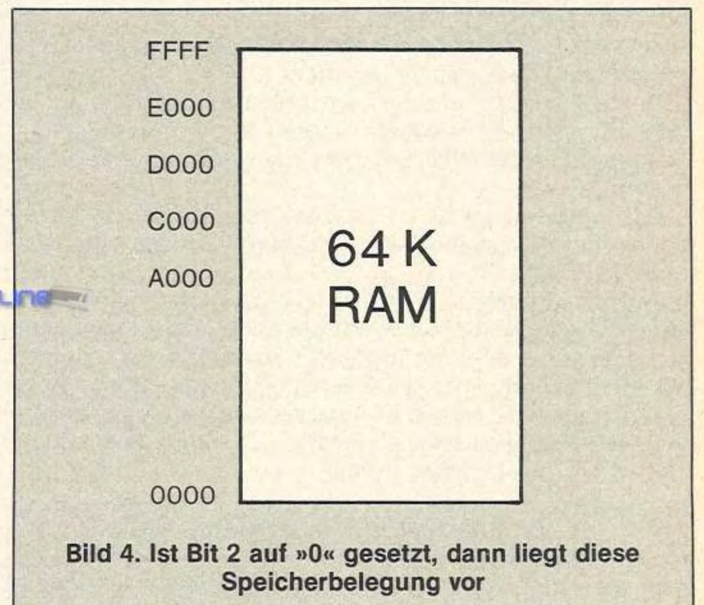


Bild 4. Ist Bit 2 auf »0« gesetzt, dann liegt diese Speicherbelegung vor

64ER ONLINE

Natürlich läßt sich der Zeichensatz auch in jeden beliebigen RAM-Bereich verschieben. Dazu muß nur der Variablen »NACH« ein anderer Wert zugewiesen werden.

Wird der oben stehende Quelltext mit »Hypra-Ass« assembliert, so kopiert der Befehl SYS 49152 den Zeichensatz an die Adresse, dessen Wert in der Variablen »NACH« abgelegt wurde.

So, jetzt ist der Zeichensatz kopiert aber, noch nicht aktiviert. Dazu muß dem Videocontroller die neue Startadresse des Zeichensatzes mitgeteilt werden.

Im Videocontroller existiert das Register 24 bei Adresse $D018 oder dezimal 53272. Dieses Register ist für die Lage des Zeichensatzes und des Video-RAMs verantwortlich. Es besteht, wie alle anderen Register auch, aus 8 Bit. Die Bits 1, 2 und 3 bestimmen die Lage des Zeichensatzes und die Bits 4, 5, 6 und 7 die Lage des Video-RAMs. Außerdem muß noch die 16 KByte-Bank, die der Videocontroller »sehen« soll, angewählt werden. Dazu dient die CIA2, und zwar das Register 0. Die entsprechende Adresse ist $DD00 beziehungsweise dezimal 56567. Die beiden niederwertigen Bits, also Bit 0 und 1 legen nun die entsprechende 16 KByte-Bank fest. Die Lage ist wie folgt definiert:

xxxx xxx00: Bank 3 = Adresse $C000 (49152) bis $FFFF (65535)

xxxx xxx01: Bank 2 = Adresse $8000 (32768) bis $BFFF (49151)

xxxx xxx10: Bank 1 = Adresse $4000 (16384) bis $7FFF (32767)

xxxx xxx11: Bank 0 = Adresse 0 bis $3FFF (16383)

Soll nun der mit dem oben stehenden Programm verschobene Zeichensatz bei $E000 aktiviert werden, geben Sie die Zeile: `POKE 56576,PEEK(56576)AND252:POKE 53272,56` ein. Der erste Teil der Zeile, bis zum Doppelpunkt, definiert die Lage der 16 KByte-Bank und der zweite Teil die Lage des Zeichensatzes und des Video-RAMs. Die Zahl, die in das Register 24 gePOKEt wird, setzt sich wie folgt zusammen:

Die Adresse des Zeichensatzes $E000 entspricht binär 1110 0000 0000 0000

Die Bits 14 und 15 werden bedingt durch die Wahl der 16 KByte Bank auf »1« gesetzt. Die Bits 13, 12 und 11, also »100«, entsprechen nun den Bits 3, 2 und 1 im Register 24 des Videocontrollers. Das Bit 0 ist nicht belegt.

Die Adresse des Video-RAMs $CC00 entspricht binär 1100 1100 0000 0000

Die Bits 14 und 15 werden wieder entsprechend auf 1 gesetzt. Die nun folgenden 4 Bits 13, 12, 11 und 10, also »0011« tauchen wieder als Bit 7, 6, 5, 4 im Register 24 des Videocontrollers auf.

Setzt man die Bits für den Zeichensatz und für das Video-RAM zusammen, ergibt sich die binäre Zahlenkombination

0011 100x

Das »x« steht für beliebig. Hier kann entweder eine »1« oder »0« eingesetzt werden. Daraus folgt, daß in das Register 24 entweder hexadezimal 38 oder 39 beziehungsweise dezimal 56 oder 57 gePOKEt werden muß.

Wenn Sie voller Erwartung die oben stehenden POKEs eingegeben haben, waren Sie sicherlich enttäuscht; denn der Cursor war weder zu erkennen, noch ließ sich irgend etwas auf dem Bildschirm darstellen. Dem Betriebssystem fehlte nämlich noch die Information, an welcher Stelle im RAM der Bildschirm zu finden ist. Aber auch dafür existiert ein POKE. Ergänzen Sie die oben stehenden POKEs zu

```
POKE 56576,PEEK(56576)AND252:POKE 53272,56:POKE
648,204 <RETURN>
```

Die Zahl, die in die Speicherzelle 648 gePOKEt wird, setzt sich aus den oberen vier Bit der Adresse zusammen, bei der das Video-RAM beginnt. Also für $CC00 entsprechend $CC oder dezimal 204.

Jetzt müßte es funktionieren. Das heißt, der neue Bildschirm muß noch gelöscht werden.

Der Zeichensatz ist nun verschoben und der Computer darauf abgestimmt. Wie aber lassen sich nun die Zeichen abändern, was ja schließlich der Sinn dieses Artikels sein sollte?

Dazu müssen wir uns zuerst mit dem Aufbau eines Zeichens und dem Aufbau des Zeichensatz-ROMs beschäftigen.

Jedes einzelne Zeichen besteht aus einer 8*8 Punkt-Matrix. Alle in dieser Matrix enthaltenen Punkte können entweder gesetzt oder nicht gesetzt sein; ein nichtgesetzter Punkt entspricht einer »0« und ein gesetzter entsprechend einer »1«. Jedes Zeichen ist nun zeilenweise gespeichert zu je 8 Bit beziehungsweise einem Byte. Das soll an dem Buchstaben »A« demonstriert werden:

```
   **      0001 1000    18    024
  ****     0011 1100    3C    060
 **  **    0110 0110    66    102
 ******  = 0111 1110 =  7E =  126
 **  **    0110 0110    66    102
 **  **    0110 0110    66    102
 **  **    0110 0110    66    102
           0000 0000    00    000
             binär      hex   dez
```

Die Bytes liegen nun von oben angefangen nacheinander im Speicher, also hexadezimal:

18 3C 66 7E 66 66 66 00

Um das zu kontrollieren, versuchen Sie einmal, den Zeichensatz von $D000 nach $2000 zu verschieben und auch dort zu aktivieren. Für ganz Eilige schon hier die Lösung: Belegen Sie im oben stehenden Quelltext die Variable »NACH« mit $2000 beziehungsweise dezimal 8192 (Zeile 20). Nach dem Assemblieren mit Hypra-Ass kann der Zeichensatz mit SYS 49152 nach $2000 verschoben und mit den POKEs:

```
POKE 56576,PEEK(56576)OR3:POKE 53272,24:POKE 648,4
<RETURN>
```

aktiviert werden. Hatten Sie den Computer zwischenzeitlich mal ausgeschaltet, ist nur ein POKE 53272,24 erforderlich. Denn der erste und letzte POKE stellen den Computer wieder auf die Einschaltkonfiguration.

Setzen Sie einen Monitor (zum Beispiel SMON) ein, läßt sich die hexadezimale Darstellung des Zeichens »A« ab Adresse $2008 mit dem Befehl »M 2008« auflisten. Ändern Sie einmal den Inhalt einer Adresse zwischen $2008 und $2010 ab. Sie werden feststellen, daß sich alle »A« auf dem Bildschirm entsprechend verändern.

Warum ist das Zeichen »A« ab Adresse $2008 zu finden? Wie schon beschrieben, werden die Zeichen byteweise hintereinander im Speicher zu je 8 Byte abgelegt. Ein Blick in das Bedienungshandbuch zum C 64 wird Ihnen zeigen, daß das Zeichen »A« den Bildschirmcode »1« hat. Mit 8 multipliziert und zur Startadresse $2000 hinzugezählt ergibt das die Startadresse $2008 für das Zeichen »A«. Die Startadresse der anderen darstellbaren Zeichen läßt sich genauso ermitteln.

Für diejenigen, die gerne Spiele programmieren, gibt es noch eine weitere interessante Möglichkeit, den Multicolor-Modus. Um diesen einzuschalten, existiert im Videocontroller das Register 22 bei Adresse $D016 (53270), und zwar das Bit 4. Ist dieses Bit auf 1 gesetzt, befindet sich der Videocontroller im Multicolor-Modus. Vom Basic aus läßt sich dieser Modus mit:

```
POKE 53270,PEEK(53270)OR16 <RETURN>
```

einschalten und mit:

```
POKE 53270,PEEK(53270)AND239 <RETURN>
```

wieder ausschalten.

Versuchen Sie doch gleich einmal, den Multicolor-Modus mit der ersten der beiden Zeilen einzugeben. Sie werden nur noch unleserliche Zeichen auf dem Bildschirm entdecken. Woran liegt das nun?

Nun, wie Sie vielleicht wissen, existiert neben dem Video-RAM auch ein Farb-RAM, in dem der Computer die Farbinformationen für die Bildschirmdarstellung speichert. Dieses Farb-RAM liegt im Bereich von $D800 bis $DBFF und kann nicht verschoben werden. Da es vom Umfang her 1 KByte groß ist, kann im normalen Modus jedem Zeichen eine andere Farbe gegeben werden. Jede Bildschirmposition hat ein korrespondierendes Byte im Farb-RAM. Auf die Farbgebung der Zeichen haben aber nur die vier unteren Bits, also Bit 3, 2, 1 und 0 eine Wirkung. Dabei hat Bit 3 eine spezielle Bedeutung. Ist es gesetzt, so wird das Zeichen im Multicolor-Modus, ist es nicht gesetzt, im Normal-Modus dargestellt. Daher lassen sich Multicolor- und normale Zeichen beliebig auf dem Bildschirm mischen. Jedem Multicolor-Zeichen werden in Abhängigkeit von den im Zeichensatz gesetzten Bits drei Farben zugeordnet. Dazu wird jedes Byte in vier Bit-Paare aufgeteilt. Nehmen wir als Beispiel wieder das Zeichen »A«, dann würde das so aussehen:

```
   **      00 01 10 00
  ****     00 11 11 00
 **  **    01 10 01 10
 ******  = 01 11 11 10
 **  **    01 10 01 10
 **  **    01 10 01 10
 **  **    01 10 01 10
           00 00 00 00
```

Die Farbzuordnung der einzelnen Bit-Paare ist wie folgt definiert:

00: Hintergrundfarbe 0 = Bildschirmfarbe = Adresse = $D021 (53281)

01: Hintergrundfarbe 1 = Adresse $D022 (53282)

10: Hintergrundfarbe 2 = Adresse $D023 (53283)

11: Sind diese beiden Bits in einem Bit-Paar gesetzt, wird die Farbe gesetzt, die durch die 3 unteren Bits, also Bit 2, 1 und 0, im Farb-RAM definiert wurde.

Mit den Informationen, die Sie hier erhalten haben, dürfte die Gestaltung eines Zeichensatz-Generators kein Problem mehr sein. Lassen Sie Ihrer Kreativität freien Lauf.

(ah)

Graphic Art – die Antwort auf das Spriteproblem

Einfarb- und Mehrfarbsprites lassen sich mit diesem Programm leicht, übersichtlich und vor allem schnell erstellen, bearbeiten und speichern.

Dieser Sprite-Editor (Listing 1 und 2) besteht aus zwei Teilen: einem Maschinenspracheteil, der schnelle Routinen zur Cursorsteuerung und zum Verschieben enthält und einem Basic-Teil, der der Tastaturabfrage dient. Diese »Zweiteilung« wirkt sich aber in keiner Weise auf die Gesamtgeschwindigkeit des Programms aus. Vielmehr erspart sie beim Abtippen die Eingabe stupider Zahlenkolonnen. Das Basic-Programm baut das Menü auf und läßt die verwendeten Sprites an der richtigen Position erscheinen (Bild 1 rechts oben in den vier Kästchen). Malt man mit »Graphic Art« seine Figur, so wird sie gleich als Sprite oben in der rechten Ecke angezeigt (und zwar normal und in x-, y- und x/y-Richtung vergrößert). So hat man sein Ergebnis immer gleich vor Augen.

Der Basic-Teil (Listing 1) steuert auch noch die Disketten-Operationen. So lassen sich die eben erstellten Sprites abspeichern oder alte zur weiteren Verschönerung einlesen. Man kann den Fehlerkanal abfragen und Disketten-Befehle senden.

Im Maschinenteil sind all die Routinen untergebracht, die in Basic zu langsam wären. Dieses sind zum Beispiel die Cursorsteuerung und das eigentliche Zeichnen, der Aufbau der Zeichenmatrix und, nicht zu vergessen, die Scroll-Routinen. Mit »Graphic Art« lassen sich in Sekundenschnelle die Sprites in alle Richtungen verschieben (dieses ist sehr wichtig, wenn man sich beim Zeichnen mal um ein paar Bildschirmpositionen geirrt hat).

Die Bedienung von Graphic Art

So, dieses erst einmal zu den Möglichkeiten und zum Aufbau des Programms. Doch nun zur Bedienung von »Graphic Art«. Nachdem man den Basic-Teil gestartet hat, wird der Maschinenspracheteil (Listing 2) nachgeladen (er steht dann von 49152 -$C000- bis $C700- 50944). Anschließend erscheint das Menü im Mehrfarbmodus. Die Umschaltung in den Einfarbmodus (und wieder zurück) erfolgt durch Drücken der M-Taste. Es wird dann ein ähnliches Menü neu aufgebaut.

Nachdem Sie sich für die Art der Sprites entschieden haben, kann es ans Zeichnen gehen. Gezeichnet wird nur mit dem Joystick (PORT 2), wobei ein Druck auf den Feuerknopf einen Punkt setzt beziehungsweise wieder löscht. Auf eine zusätzliche Steuerung mit den Cursor-Tasten wurde verzichtet, da über diese Tasten das Verschieben der Sprites geregelt wird. Dieses geht nicht nur schnell, sondern damit auch in den gewohnten Richtungen besonders einfach und übersichtlich.

Die Spritefarben werden alle über die Funktionstasten gesteuert, wobei F1/F2 für die Hauptspritefarbe ($D027), F3/F4 für die Hintergrundfarbe (es erscheint ein Farbfenster direkt hinter den Sprites), F5/F6 für die erste Nebenspritefarbe ($D025 – nur wichtig für den Multicolour-Modus) und F7/F8 für die Nebenfarbe 2 ($D026) gilt.

Im Multicolour-Modus werden alle drei Farben angezeigt (Multicolour-Sprites erlauben das Zeichnen mit höchstens drei Farben). Das gleiche gilt auch für die aktuelle Zeichenfarbe. Sie kann man aus den Spritefarben 1 bis 3 durch Drücken von SHIFT und der entsprechenden Farbnummer (also 1, 2 oder 3) herauswählen. Der Zeichenfarbwechsel wird dann angezeigt.

Um im Multicolour-Modus Sprites zu erstellen, gehen Sie am besten so vor: Zuerst entscheiden Sie, welche drei Farben Ihr Sprite haben soll (mit F-Tasten). Anschließend wählen Sie die Zeichenfarbe und beginnen zu zeichnen. Achtung, wenn Sie eine der drei Farben wechseln wollen, so wird jeder Punkt, der mit dieser Farbe gezeichnet wurde, auf die neu eingestellte Farbe umgestellt! Sollte die Farbe, die geändert wird, die aktuelle Zeichenfarbe sein, so passen Sie bitte anschließend die Zeichenfarbe neu an (mit SHIFT und Farbnummer), da sich das Programm nicht merkt, welche der drei Farben die Zeichenfarbe ist.

Der Spritepuffer

Mit »Graphic Art« lassen sich acht Sprites direkt im Computer ablegen. Dem Benutzer stehen folglich acht «Sprite-Ablegebereiche« zur Verfügung, in die man durch Drücken der »1«-Taste den gerade sichtbaren Bereich hineinschreiben kann. Zurückholen läßt sich das Sprite mit der »*«-Taste. Natürlich wird das gerade sichtbare Sprite dabei überschrieben. »Graphic Art« bietet auch die Möglichkeit des Speicherns der Spritedaten auf Diskette sowie (für nicht-Floppy-Besitzer) der Ausgabe der Daten auf Bildschirm und Drucker.

Drücken Sie »A« für Ausgabe und im unteren Teil des Menüs erscheinen die drei Wahlmöglichkeiten. Drücken Sie nun »1« für Diskette, so wird das Sprite, nachdem ihm ein Name gegeben wurde, auf Diskette als SEQ-Datei gespeichert und läßt sich nun leicht in eigene Programme nachladen (so erspart man sich die DATA-Zeilen – aber Achtung: die Farben werden nicht mit gespeichert! Sie müssen nach Laden der Daten in die entsprechenden Farbregister gePOKEt werden).

Wer lieber DATAs mag oder kein Disketten-Laufwerk hat, der kann sich die Spritedaten ausdrucken oder auf dem Bildschirm ausgeben lassen. In diesem Fall wird man noch gefragt, ob das Programm beendet werden soll. Wenn ja, so ist der Editor gelöscht. Dafür stehen nun in den Zeilen 1 bis 6 die Spritedaten. Diese können nun mit einem MERGE in Ihr Programm übernommen werden.

Natürlich ist es auch möglich, gespeicherte Sprites wieder zu laden und weiter zu bearbeiten. Dazu drückt man die ←-Taste.

Floppy-Besitzer werden sich freuen, denn »Graphic Art« läßt auch die Eingabe von Disketten-Befehlen zu. So kann man eine Diskette neu formatieren, ein File löschen und so weiter. Dieses und die Abfrage des Fehlerkanals wird durch den Druck auf die Klammeraffen-Taste ermöglicht (eine Anmerkung noch: alle Untermenüs können durch Drücken der RETURN-Taste verlassen werden).

Damit ist die Bedienung der Aus- und Eingabegeräte erklärt. Ich will nun auf das Zeichnen selbst mit »Graphic Art« noch ein wenig genauer eingehen.

Normalerweise wird durch Drücken des Feuerknopfes ein Punkt gesetzt oder gelöscht. Durch ständiges Drücken kann das Ziehen einer horizontalen Linie erreicht werden. Ist jedoch das Zeichnen einer vertikalen Linie erwünscht, so schaltet man den DRAW-Modus ein (durch Drücken von D-an/Shift D-aus). Nun wird jeder Punkt, über den man mit dem Cursor fährt, gesetzt (oder gelöscht). So ist ein Zeichnen von Linien sehr einfach.

So wird gezeichnet

Manchmal ist das Löschen aber dabei nicht erwünscht (zum Beispiel wenn eine Fläche ausgefüllt werden soll). »Graphic Art« bietet hierfür die Möglichkeit, mit dem Feuerknopf (beziehungsweise im Draw-Modus bei Cursorbewegung) nur zu setzen und nicht zu löschen. Dieses Nur-Setzen wird im Menü durch den S-Modus gekennzeichnet – im Gegensatz dazu der Setzen/Löschen-(=S/L-)Modus. Durch Drücken der S-Taste kann man sich den gewünschten Modus auswählen. Der S-Modus funktioniert aber nur bei der Erstellung der Einfarbsprites. Allerdings ist es möglich, durch Drücken der SPACE-Taste doch noch einen Punkt zu löschen.
Und nun noch ein paar Kleinigkeiten:
Durch Drücken der Tasten 1 bis 9 wird eine Geschwindigkeitssänderung des Cursors erreicht. Auch eine Reverse-Funktion ist in »Graphic Art« mit eingebaut. Durch Betätigen der 0-Taste erscheint das Sprite dann reverse. Dieses ist sehr angenehm und erspart viel Arbeit, wenn man ein fast ganz ausgefülltes Sprite malen will. Übrigens lassen sich einige Cursor-Bewegungen auch von der Tastatur aus steuern. So wird nach Drücken von CLR das Sprite gelöscht, ein HOME läßt den Cursor links oben in der Ecke erscheinen und ein RETURN setzt ihn an den Anfang der nächsten Zeile.
Eine Sache noch, die besonders den Multicolour-Modus betrifft. Dabei kann es vorkommen, daß die Cursor-Lupe, die eigentlich gelb ist, nicht mehr erkannt werden kann, wenn das gerade bearbeitete Sprite ebenfalls zum Teil die gleiche Farbe besitzt. Um dennoch immer über die genaue Zeichenposition im klaren zu sein, kann durch Betätigen der +/–Tasten die Cursorfarbe geändert werden.
Und nun zu guter Letzt noch zwei Hinweise zum Abtippen. Zum einen können die REMs alle weggelassen werden. Zum anderen ist in einige Basic-Zeilen so viel reingepreßt worden, daß die Befehle zum Teil verkürzt eingegeben werden müssen (für PRINT also »?« und so weiter – schauen Sie eventuell im Handbuch nach!). Vor dem Starten unbedingt beide Programme speichern!!!
Ganz zum Schluß noch ein Tip: »Graphic Art« berechnet zwar Ihre Sprites, wie diese dann aber gesteuert werden, kann es nicht erraten.

(Andreas Holz/ah)

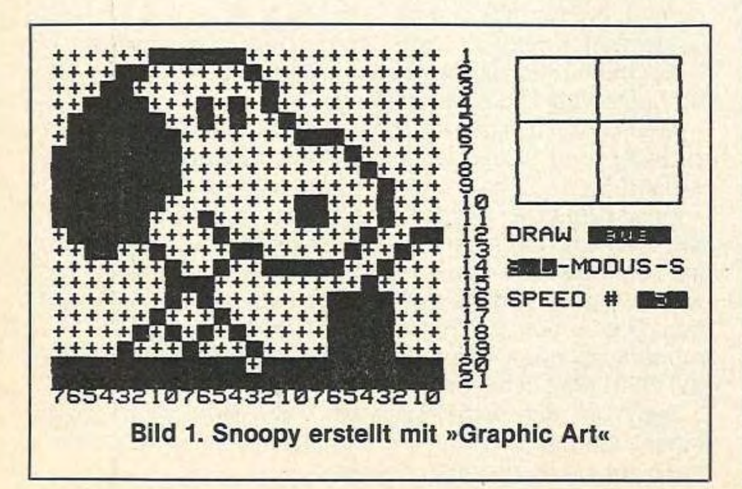


Bild 1. Snoopy erstellt mit »Graphic Art«

```
5 POKE 56,61:POKE 55,192                       <078>
10 POKE 53280,0:POKE 53281,0:POKE 650,128:
   POKE 53272,23:PRINT"{CLR}"                  <014>
11 ::                                          <196>
12 IF A=0 THEN A=1:LOAD"MT-GRAFIK-ART",8,1     <077>
99 ::                                          <028>
100 REM+++++++++++++++++++++++++++++++         <101>
101 REM+                             +         <174>
102 REM+      GRAFIK-ART             +         <068>
103 REM+                             +         <176>
104 REM+            ANDREAS A. HOLZ  +         <200>
105 REM+            ZIKADENWEG 15    +         <078>
106 REM+ [3017728] 1   BERLIN 19     +         <160>
107 REM+                             +         <180>
108 REM+++++++++++++++++++++++++++++++         <109>
109 ::                                         <038>
120 FOR X=704 TO 766:READ A:POKE X,A:NEXT:
    REM SINGLE CUSOUR [11. BLOCK]              <063>
130 FOR X=896 TO 958:READ A:POKE X,A:NEXT:
    REM MULTI  CUSOUR [14. BLOCK]              <036>
135 FOR X=1 TO 9:READ SP(X):NEXT:REM +++++
    SPEED-DATAS                                <099>
140 V=53248:POKE V+21,143:POKE V+14,0:POKE
     V+15,29:POKE V+46,7:POKE V+29,134:POK
    E V+23,140                                 <113>
150 V$="{BLUE}7654321{LIG.BLUE}0":L$="{39S
    PACE}":POKE 198,0                          <083>
155 GOSUB 710:GOSUB 830:SYS 49702:REM BILD
    SCHIRMAUFBAU+SPRITE LOESCHEN [13.BLOCK
    ]                                          <127>
170 POKE 2040,13:POKE V+16,6:POKE V,255:PO
    KE V+1,51:POKE V+39,14:REM+  SPRITE 0      <032>
180 POKE 2041,13:POKE V+2,28:POKE V+3,51:P
    OKE V+40,14:REM+++++++++++  SPRITE 1       <199>
190 POKE 2042,13:POKE V+4,28:POKE V+5,80:P
    OKE V+41,14:REM+++++++++++  SPRITE 2       <212>
200 POKE 2043,13:POKE V+6,255:POKE V+7,80:
    POKE V+42,14:REM++++++++++  SPRITE 3       <212>
298 ::                                         <229>
299 POKE 198,0                                 <209>
300 SYS 49152                                  <104>
305 GET A$:IF A$=""GOTO 300                    <018>
310 IF A$="{F3}"THEN SYS 49682:GOTO 300        <124>
320 IF A$="{F4}"THEN SYS 49692:GOTO 300        <200>
322 IF A$="{RIGHT}"THEN SYS 50559:GOTO 300 <237>
324 IF A$="{LEFT}"THEN SYS 50542:GOTO 300      <108>
326 IF A$="{DOWN}"THEN SYS 49879:GOTO 300      <084>
328 IF A$="{UP}"THEN SYS 49931:GOTO 300        <177>
350 IF A$="D"THEN SYS 49778:GOSUB 800:GOTO
     299                                       <238>
355 IF A$="D"THEN SYS 49794:GOSUB 800:GOTO
     299                                       <178>
360 IF A$="{CLR}"THEN SYS 49702:GOTO 299       <038>
370 IF A$="{HOME}"THEN POKE V+14,0:POKE V+
    15,29:GOTO 299                             <032>
380 IF A$=CHR$(13)THEN IF PEEK(V+15)<>189
    THEN POKE V+14,0:POKE V+15,PEEK(V+15)+
    8:GOTO 299                                 <185>
390 IF A$="S"THEN SYS 49821:GOSUB 800:GOTO
     299                                       <249>
392 IF A$="{F1}"THEN SYS 49720:GOTO 300        <234>
393 IF A$="{F2}"THEN SYS 49735:GOTO 300        <175>
394 IF A$="{F5}"THEN SYS 50502:GOTO 300        <166>
395 IF A$="{F6}"THEN SYS 50522:GOTO 300        <041>
396 IF A$="{F7}"THEN SYS 50512:GOTO 300        <105>
397 IF A$="{F8}"THEN SYS 50532:GOTO 300        <236>
400 IF A$=" "THEN IF PEEK(49405)=176 THEN
    SYS 49844:GOTO 300                         <027>
410 IF A$="0"THEN FOR X=832 TO 894:POKE X,
    255-PEEK(X):NEXT:SYS 49548:GOTO 299        <103>
490 IF A$="M"THEN SYS 50065:GOSUB 710:GOSU
    B 830:SYS 49548:GOTO 299                   <142>
495 IF A$="A"THEN GOSUB 1010:GOTO 300          <155>
497 IF A$="@"GOTO 3005                         <150>
500 IF A$="!"THEN IF PEEK(49236)=15 THEN P
    OKE 49997,128:POKE 49996,PEEK(V+39):GO
    SUB 760:GOTO 299                           <012>
510 IF A$=CHR$(34)THEN IF PEEK(49236)=15 T
    HEN POKE 49997,64:POKE 49996,PEEK(V+37
    ):GOSUB 760:GOTO 299                       <217>
520 IF A$="#"THEN IF PEEK(49236)=15 THEN P
    OKE 49997,192:POKE 49996,PEEK(V+38):GO
    SUB 760:GOTO 299                           <096>
525 IF A$="←"THEN GOSUB 4000:GOTO 299          <044>
```

Listing 1. Basic-Listing zu »Graphic Art«

64er online

```
527 IF ASC(A$)>48 AND ASC(A$)<58 THEN POKE
    1661,128+ASC(A$):POKE 49294,SP(ASC(A$
    )-48):GOTO 300                          <075>
528 IF A$="+"THEN IF PEEK(V+46)<>255 THEN
    POKE V+46,PEEK(V+46)+1:GOTO 299         <226>
529 IF A$="-"THEN IF PEEK(V+46)<>0 THEN PO
    KE V+46,PEEK(V+46)-1:GOTO 299           <057>
535 IF A$="↑"GOTO 7010                      <077>
540 IF A$="*"GOTO 7050                      <120>
550 GOTO 300                                <242>
700 REM++++++++++++++++++++++++             <242>
701 REM+                      +             <010>
702 REM+  BILDSCHIRMAUFBAU    +             <018>
703 REM+                      +             <012>
704 REM++++++++++++++++++++++++             <246>
705 ::                                      <126>
710 POKE V+14,0:POKE V+15,29                <088>
715 PRINT"{CLR}";:IF PEEK(49236)=15 GOTO 7
    40                                      <093>
720 FOR X=1 TO 21:PRINT"{RED}+++++++++++++
    +++++++++++{BLUE}"X:NEXT:FOR X=0 TO 2:
    PRINT V$;:NEXT                          <057>
730 POKE 2047,11:POKE V+28,0:GOTO 800       <039>
740 FOR X=1 TO 21:PRINT"{RED}DFDFDFDFDFDFDFD
    FDFDFDFDFDF{BLUE}"X:NEXT:FOR X=0 TO 2:
    PRINT V$;:NEXT                          <074>
750 POKE 2047,14:POKE V+28,15               <078>
760 POKE 214,17:POKE 211,28:SYS 58640:PRIN
    T"{CYAN}ZEICHEN= ";:POKE 646,PEEK(4999
    6):PRINT"{RVSON,2SPACE,DOWN,2LEFT,2SPA
    CE,RVOFF}"                              <198>
765 POKE 214,18:POKE 211,31:SYS 58640:PRIN
    T"{CYAN}FARBE"                          <149>
770 POKE 214,20:POKE 211,28:SYS 58640:PRIN
    T"{LIG.RED}FARBE #1 ";:POKE 646,PEEK(5
    3287):PRINT"{RVSON,2SPACE,RVOFF}"       <135>
780 POKE 214,21:POKE 211,28:SYS 58640:PRIN
    T"{RED}FARBE #2 ";:POKE 646,PEEK(53285
    ):PRINT"{RVSON,2SPACE,RVOFF}"           <190>
790 POKE 214,22:POKE 211,28:SYS 58640:PRIN
    T"{PURPLE}FARBE #3 ";:POKE 646,PEEK(53
    286):PRINT"{RVSON,2SPACE,RVOFF}"        <211>
800 POKE 214,11:POKE 211,28:SYS 58640:PRIN
    T"{BLUE}DRAW{SPACE,RVSON,SPACE}A";      <154>
805 IF PEEK(49280)<>234 GOTO 808            <120>
806 PRINT"N{2SPACE,RVOFF}":GOTO 810         <188>
808 PRINT"US{SPACE,RVOFF}"                  <025>
810 POKE 214,13:POKE 211,28:SYS 58640:PRIN
    T"{LIG.BLUE}S/L-MODUS-S":IF PEEK(49404
    )=43 THEN POKE 1582,128+PEEK(1582)      <033>
820 IF PEEK(49404)<>43 THEN FOR X=1572 TO
    1574:POKE X,128+PEEK(X):NEXT:RETURN     <187>
825 RETURN                                  <121>
830 POKE 214,15:POKE 211,28:SYS 58640:PRIN
    T"{GREEN}SPEED #{SPACE,RVSON,SPACE}5{S
    PACE,RVOFF}":POKE 49294,64:RETURN       <238>
839 ::                                      <006>
840 REM++++  SINGLE-SPRITE-DATA'S  ++++     <017>
845 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,7,,,8,
    128,,16,64,,16,64,,8,192,,7,224,,,48    <168>
850 DATA,,24,,,,,,,,,,,,                    <035>
900 :                                       <114>
905 REM++++  MULTI -SPRITE-DATA'S  ++++     <039>
910 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,31,
    252,,32,2,,32,2,,32,2,,32,2,,31,252,,   <192>
915 DATA,,,,,,,,,,,,                        <087>
920 :                                       <134>
925 REM++++  SPEED - DATAS ++++++++++++     <168>
930 DATA 80,75,68,64,58,48,35,25,17         <193>
935 ::                                      <102>
1000 REM+++++++++++++++++++++++             <116>
1001 REM+                     +             <056>
1002 REM+      AUSGABE        +             <239>
1003 REM+                     +             <058>
1004 REM+++++++++++++++++++++++             <120>
1005 :                                      <219>
1010 POKE 198,0:POKE 214,23:POKE 211,0:SYS
     58640:PRINT"{LIG.BLUE}AUSGABE..."      <124>
1020 PRINT"1. DISK{3SPACE}2. DRUCKER{3SPAC
     E}3. SCREEN";                          <172>
1030 GET A$:IF A$="1"GOTO 2001              <204>
1035 IF A$="2"GOTO 1050                     <087>
1040 IF A$="3"THEN GOSUB 1055:GOTO 1080     <187>
1042 IF A$=CHR$(13)THEN GOSUB 1095:GOTO 30
     0                                      <126>
1045 GOTO 1030                              <065>
1050 OPEN 4,4:CMD 4:GOSUB 1060:PRINT#4:CLO
     SE 4:GOSUB 1095:RETURN                 <223>
1055 PRINT"{CLR}";:POKE V+21,0              <040>
1060 FOR X=0 TO 4:PRINT X+1"DATA";:FOR X1=
     0 TO 10:PRINT PEEK(832+X1+X*12)"{LEFT
     },";                                   <197>
1065 NEXT:PRINT PEEK(832+X1+X*12)           <051>
1070 NEXT:PRINT X+1"DATA";:FOR X=892 TO 89
     3:PRINT PEEK(X)",";:NEXT:PRINT PEEK(8
     94):RETURN                             <179>
1080 INPUT"{2DOWN}PROGRAMM ENDE";A$:IF MID
     $(A$,1,1)="J"GOTO 5000                 <209>
1090 GOSUB 710:GOSUB 830:POKE V+21,143:SYS
     49548:RETURN                           <172>
1095 FOR X=0 TO 1:POKE 214,23+X:POKE 211,0
     :SYS 58640:PRINT L$;:NEXT:RETURN       <146>
1097 POKE 53280,2:POKE 53265,PEEK(53265)AN
     D 239:RETURN                           <201>
1098 POKE 53265,PEEK(53265)OR 16:POKE 5328
     0,0:RETURN                             <189>
1099 OPEN 2,8,2,N$+",S,W":RETURN            <132>
1100 ::                                     <013>
1101 CLOSE 2:GOSUB 1099:GOSUB 1095:SYS 506
     07:GOSUB 1098:WAIT 198,1:GOSUB 1095:S
     YS 65511:RETURN                        <163>
2000 ::                                     <151>
2001 GOSUB 1095:GOSUB 4101:IF N$=""THEN GO
     SUB 1095:GOTO 300                      <156>
2010 GOSUB 1097:GOSUB 1099:OPEN 15,8,15:IN
     PUT#15,A$:IF A$<>"00"GOTO 1101         <214>
2015 FOR X=832 TO 832+62:PRINT#2,PEEK(X)    <239>
2020 NEXT                                   <252>
2025 CLOSE 2:CLOSE 15:GOSUB 1095:GOSUB 109
     8:RETURN                               <120>
3000 REM+++++++++++++++++++++++++++         <112>
3001 REM+                         +         <024>
3002 REM+    DISK......           +         <113>
3003 REM+                         +         <026>
3004 REM+++++++++++++++++++++++++++         <116>
3005 POKE 214,23:POKE 211,0:SYS 58640:PRIN
     T"{LIG.BLUE}DISK...":PRINT"1. DOS{3SP
     ACE}2. DISK-STATUS";                   <136>
3010 GET A$:IF A$="1"GOTO 3050              <217>
3015 IF A$="2"GOTO 3070                     <073>
3016 IF A$=CHR$(13)THEN GOSUB 1095:GOTO 29
     9                                      <235>
3020 GOTO 3010                              <214>
3050 GOSUB 1095:POKE 214,23:POKE 211,0:SYS
     58640:GOSUB 4100:IF N$=""THEN GOSUB
     1095:GOTO 300                          <137>
3055 OPEN 1,8,15,N$:CLOSE 1:GOSUB 1095:GOT
     O 300                                  <028>
3070 GOSUB 1095:SYS 50709:WAIT 198,1:GOSUB
     1095:POKE 198,0:GOTO 299               <202>
3100 REM+++++++++++++++++++++++++++         <214>
3101 REM+                         +         <126>
3102 REM+    EINGABE              +         <179>
3103 REM+                         +         <128>
3104 REM+++++++++++++++++++++++++++         <214>
3998 OPEN 2,8,2,N$+",S,R":RETURN            <232>
3999 CLOSE 2:GOSUB 3998:GOSUB 1095:SYS 506
     07:GOSUB 1098:WAIT 198,1:GOSUB 1095:S
     YS 65511:RETURN                        <013>
4000 GOSUB 4101:IF N$=""THEN GOSUB 1095:GO
     TO 300                                 <126>
4001 GOSUB 1097:GOSUB 3998:OPEN 15,8,15:IN
     PUT#15,A$:IF A$<>"00"GOTO 3999         <059>
4005 FOR X=832 TO 832+62:INPUT#2,A:POKE X,
     A:NEXT                                 <078>
4010 CLOSE 2:CLOSE 15:GOSUB 1095:SYS 49548
     :GOSUB 1098:RETURN                     <067>
4091 ::                                     <210>
4092 REM +++   GET  NAME   +++              <178>
4093 ::                                     <212>
4100 N$="":POKE 214,23:POKE 211,0:SYS 5864
     0:PRINT"{LIG.BLUE}BEFEHL? ";:N=29:GOT
     O 4105                                 <123>
4101 N$="":POKE 214,23:POKE 211,0:SYS 5864
     0:PRINT"{LIG.BLUE}FILENAME? ";:N=16    <123>
4105 A$="":POKE 198,0:POKE 204,0            <232>
4110 GET A$:IF A$=""THEN 4110               <255>
4114 IF A$=CHR$(13)THEN POKE 204,1:PRINT"
     ";:RETURN                              <133>
4120 IF A$=CHR$(20)THEN IF N$<>""THEN POKE
```

Listing 1. Basic-Listing zu »Graphic Art«. (Fortsetzung)

```
      204,1:PRINT"{LEFT,2SPACE,2LEFT}";:N$
      =LEFT$(N$,LEN(N$)-1):GOTO 4105          <035>
4125 IF N$=""THEN IF A$=CHR$(20)THEN 4110     <062>
4130 IF LEN(N$)=N THEN 4110                   <072>
4135 POKE 204,1:PRINT A$;:N$=N$+A$:GOTO 41
     05                                       <176>
5000 POKE 631,19:FOR X=632 TO 632+6:POKE X
     ,13:NEXT:POKE 198,7:NEW                  <217>
7003 ::                                       <074>
7004 REM **** SPRITE-SPEICHER                 <175>
7005 ::                                       <076>
7010 POKE 214,23:POKE 211,0:SYS 58640:PRIN
     T"{LIG.BLUE}SPRITE SPEICHERN"            <128>
7015 N$="":N=1:PRINT"SPRITENR. (1-8)? ";:G
     OSUB 4105:IF N$=""THEN GOTO 7065         <125>
7016 X=VAL(N$):IF X<1 OR X>8 GOTO 7065        <246>
7020 FOR X1=0 TO 62:POKE 15808+(64*(X-1))+
     X1,PEEK(832+X1):NEXT:GOSUB 1095:GOTO
     299                                      <052>
7030 ::                                       <101>
7050 POKE 214,23:POKE 211,0:SYS 58640:PRIN
     T"{LIG.BLUE}SPRITE EINLESEN"             <006>
7055 N$="":N=1:PRINT"SPRITENR. (1-8)? ";:G
     OSUB 4105:IF N$=""THEN GOSUB 1095:GOT
     O 299                                    <115>
7060 X=VAL(N$):FOR X1=0 TO 62:POKE 832+X1,
     PEEK(15808+(64*(X-1))+X1):NEXT           <054>
7065 GOSUB 1095:SYS 49548:GOTO 299            <217>
```

Listing 1. Basic-Listing zu »Graphic Art«. (Schluß)

```
programm : mt-grafik-art   c000 c700
--------------------------------------

c000 : a5 cb c9 40 f0 01 60 ad   f9
c008 : 00 dc aa a8 29 01 f0 1a   c9
c010 : 8a 29 02 f0 26 98 29 04   a1
c018 : f0 32 ad 00 dc aa 29 08   65
c020 : f0 44 8a 29 10 f0 59 4c   81
c028 : 00 c0 ad 0f d0 c9 1d f0   87
c030 : 52 38 e9 08 38 8d 0f d0   e8
c038 : 4c 83 c0 ad 0f d0 c9 bd   46
c040 : f0 41 18 69 08 18 8d 0f   9a
c048 : d0 4c 83 c0 ad 0e d0 c9   59
c050 : 00 f0 30 c9 0f d0 05 a9   ec
c058 : 00 4c 60 c0 38 e9 10 38   32
c060 : 8d 0e d0 4c 83 c0 ad 0e   c3
c068 : d0 c9 af f0 16 c9 00 d0   78
c070 : 05 a9 0f 4c 7a c0 18 69   78
c078 : 10 18 8d 0e d0 4c 83 c0   b8
c080 : 4c 94 c0 a2 00 a0 00 c8   31
c088 : c0 ff d0 fb e8 e0 40 d0   34
c090 : f4 4c 07 c0 ae 0e d0 e8   f5
c098 : 8a a2 00 c9 08 90 0d 38   56
c0a0 : e9 08 38 c9 08 90 04 e8   bc
c0a8 : 4c 9f c0 e8 ac 0f d0 c8   29
c0b0 : 98 a0 00 38 e9 08 38 c9   f3
c0b8 : 08 90 04 c8 4c b3 c0 88   99
c0c0 : 88 a9 04 85 fb a9 00 85   e7
c0c8 : fa 4c ce c0 06 06 8e cc   18
c0d0 : c0 8c cd c0 18 6d cc c0   03
c0d8 : 85 fa a9 00 65 fb 18 85   e6
c0e0 : fb a2 00 a5 fa 18 6d cd   a3
c0e8 : c0 85 fa a9 00 65 fb 85   85
c0f0 : fb 18 e8 e0 28 d0 ec a0   4b
c0f8 : 00 b1 fa c9 a0 d0 07 4c   0e
c100 : 5c c4 fa 4c 40 c1 4c 1c   82
c108 : c4 fa 4c 0e c1 06 ad cc   bb
c110 : c0 a2 00 c9 08 90 08 38   f0
c118 : e9 08 38 e8 4c 13 c1 8d   b0
c120 : 0d c1 8a a0 00 18 6d cd   d7
c128 : c0 18 c8 c0 03 d0 f6 aa   26
c130 : ac 0d c1 60 71 c1 1d 40   f9
c138 : 03 9d 40 03 4c 65 c2 07   83
c140 : ad cc c0 a2 00 c9 08 90   67
c148 : 08 38 e9 08 38 e8 4c 45   6e
c150 : c1 8d 3f c1 8a a0 00 18   be
c158 : 6d cd c0 18 c8 c0 03 d0   1f
c160 : f6 aa ac 3f c1 b9 79 c1   12
c168 : 3d 40 03 9d 40 03 60 65   a2
c170 : c2 80 40 20 10 08 04 02   dc
c178 : 01 7f bf df ef f7 fb fd   cf
c180 : fe a9 00 a2 0a 9d 76 02   13
c188 : ca d0 fa 60 4c 89 c4 04   b1
c190 : 85 fa 84 fb a9 40 a0 03   58
c198 : 85 fc 84 fd a2 00 a0 00   29
c1a0 : b1 fc 0a 85 02 90 07 a9   17
c1a8 : a0 91 fa 4c b2 c1 a9 2b   8f
c1b0 : 91 fa c8 a5 02 c0 08 d0   8d
c1b8 : e9 e6 fc 18 a9 08 65 fa   bd
c1c0 : 85 fa a9 00 65 fb 85 fb   71
c1c8 : e8 e0 03 d0 d1 a9 10 18   d6
c1d0 : 65 fa 85 fa a9 00 65 fb   9b
c1d8 : 85 fb a5 fc c9 7f d0 bc   b9
c1e0 : 60 ad 0f c2 8d 86 02 a2   8d
c1e8 : 00 a0 1c 18 20 f0 ff ea   a2
c1f0 : ea 20 ff c1 c8 c0 27 d0   f3
c1f8 : f2 e8 e0 0a d0 eb 60 a9   19
c200 : 12 20 d2 ff a9 20 20 d2   99
c208 : ff a9 92 20 d2 ff 60 ad   8e
c210 : ea ea ee 0f c2 20 e1 c1   45
c218 : 20 81 c1 60 ce 0f c2 20   26
c220 : e1 c1 20 81 c1 60 a9 00   e0
c228 : aa 9d 40 03 e8 e0 3f d0   45
c230 : f8 20 8c c1 20 81 c1 60   69
c238 : ee 27 d0 ee 28 d0 ee 29   e3
c240 : d0 ee 2a d0 4c 53 c2 ce   34
c248 : 27 d0 ce 28 d0 ce 29 d0   5a
c250 : ce 2a d0 a2 00 a0 00 c8   52
c258 : c0 ff d0 fb e8 e0 01 d0   07
c260 : f4 4c 45 c3 ea ad 91 c0   28
c268 : c9 ea d0 03 4c 07 c0 4c   d3
c270 : 66 c0 a2 00 a9 ea 9d 91   6a
c278 : c0 9d 80 c0 e8 e0 03 d0   82
c280 : f5 60 a9 4c a2 07 a0 c0   00
c288 : 8d 91 c0 8e 92 c0 8c 93   68
c290 : c0 8d 80 c0 8c 82 c0 a2   74
c298 : 94 8e 81 c0 60 a2 00 bd   82
c2a0 : fc c0 bc b2 c2 9d b2 c2   eb
c2a8 : 98 9d fc c0 e8 e0 02 d0   a5
c2b0 : ee 60 2b b0 a9 60 8d 3c   fb
c2b8 : c1 8d 6e c1 8d 7d c0 20   1c
c2c0 : 9d c2 20 94 c0 20 9d c2   62
c2c8 : 20 66 c0 a9 4c 8d 3c c1   26
c2d0 : 8d 6e c1 8d 7d c0 60 a0   57
c2d8 : 3c a2 00 b9 40 03 95 fa   05
c2e0 : c8 e8 e0 03 d0 f5 a0 00   f4
c2e8 : a2 00 b9 40 03 85 fd b5   c0
c2f0 : fa 99 40 03 a5 fd 95 fa   be
c2f8 : e8 c8 e0 03 d0 ec a2 00   dc
c300 : c0 3f d0 e6 20 8c c1 20   1e
c308 : 81 c1 60 a0 00 a2 00 b9   1f
c310 : 40 03 95 fa c8 e8 e0 03   f4
c318 : d0 f5 a0 3c a2 00 b9 40   24
c320 : 03 85 fd b5 fa 99 [illegible] 03   9f
c328 : a5 fd 95 fa e8 c8 e0 03   ef
c330 : d0 ec 88 88 88 88 88 88   a9
c338 : a2 00 c0 fd d0 e0 20 8c   78
c340 : c1 20 81 c1 60 20 81 c1   3a
c348 : 20 90 c5 60 fe 80 ea ea   a3
c350 : a2 00 bd 40 03 0a e8 e8   60
c358 : 3e 40 03 ca 3e 40 03 ca   58
c360 : 3e 40 03 e8 e8 e8 e0 3f   74
c368 : 90 e8 20 81 c1 60 ea ea   45
c370 : a2 02 bd 40 03 4a ca ca   ce
c378 : 7e 40 03 e8 7e 40 03 e8   bc
c380 : 7e 40 03 e8 e8 e8 e0 3f   d4
c388 : 90 e8 20 81 c1 60 ea ea   65
c390 : ea a2 0f ec 54 c0 f0 40   bc
c398 : 8e 54 c0 8e 72 c0 e8 8e   40
c3a0 : 5e c0 8e 78 c0 a9 5c a2   21
c3a8 : 4c a0 c4 8d 00 c1 8e ff   6f
c3b0 : c0 8e 06 c1 8e 8c c1 8c   de
c3b8 : 01 c1 8c 08 c1 8c 8e c1   fc
c3c0 : a9 60 8d 33 c1 8d 6e c1   29
c3c8 : a9 89 8d 8d c1 a2 af a0   7c
c3d0 : 1c 8c 07 c1 8e 6a c0 60   2c
c3d8 : a2 07 8e 54 c0 8e 72 c0   f8
c3e0 : e8 8e 5e c0 8e 78 c0 a9   c2
c3e8 : a9 a0 2b a2 91 8d ff c0   07
c3f0 : 8d 06 c1 8d 8c c1 8c 00   ab
c3f8 : c1 8e 01 c1 8e 08 c1 a9   fc
c400 : a0 8d 07 c1 8d 8e c1 a9   08
c408 : 00 8d 8d c1 a2 b9 8e 33   03
c410 : c1 ca ca 8e 6a c0 a9 4c   a7
c418 : 8d 6e c1 60 a0 01 a9 a0   53
c420 : 91 fa a9 d4 18 65 fb 18   00
c428 : 85 fb ad 4c c3 91 fa a5   a0
c430 : fb 38 e9 d4 38 85 fb 88   0d
c438 : ea ea c0 00 f0 e0 20 0e   7a
c440 : c1 ad 4d c3 c0 02 90 07   10
c448 : 88 88 4a 4a 4c 44 c4 1d   24
c450 : 40 03 9d 40 03 4c 65 c2   2f
c458 : 00 02 ea ea 20 40 c1 ee   5a
c460 : cc c0 20 40 c1 a0 01 a9   15
c468 : 7b 91 fa a9 d4 18 65 fb   3b
c470 : 18 85 fb a9 02 91 fa a5   63
c478 : fb 38 e9 d4 38 85 fb 88   55
c480 : d0 04 a9 6c d0 e3 4c 65   72
c488 : c2 a9 00 a0 04 85 fa 84   94
c490 : fb a9 40 a0 03 85 fc 84   dd
c498 : fd a2 00 8e 58 c4 a0 00   e6
c4a0 : b1 fc 0a 2e 58 c4 18 20   64
c4a8 : 3c c5 d0 15 a9 6c 91 fa   d8
c4b0 : c8 a9 7b 91 fa a9 02 8d   7e
c4b8 : 59 c4 20 1f c5 c8 4c ec   0d
c4c0 : c4 c9 01 d0 09 ad 25 d0   f7
c4c8 : 8d 59 c4 4c e1 c4 c9 02   2c
c4d0 : d0 09 ad 27 d0 8d 59 c4   dd
c4d8 : 4c e1 c4 ad 26 d0 8d 59   cd
c4e0 : c4 a9 a0 91 fa c8 91 fa   05
c4e8 : 20 1f c5 c8 a9 00 8d 58   a4
c4f0 : c4 a5 02 c0 08 d0 ab e6   a3
c4f8 : fc a9 08 18 65 fa 85 fa   08
c500 : a9 00 65 fb 85 fb e8 e0   20
c508 : 03 d0 93 a9 10 18 65 fa   db
c510 : 85 fa a9 00 65 fb 85 fb   c1
c518 : a5 fc c9 7f d0 1b 60 88   16
c520 : a9 d4 18 65 fb 18 85 fb   75
c528 : ad 59 c4 91 fa c8 91 fa   17
c530 : a5 fb 38 e9 d4 38 85 fb   3b
c538 : 60 4c 99 c4 0a 2e 58 c4   ba
c540 : 85 02 ad 58 c4 60 ee 25   92
c548 : d0 20 81 c1 20 90 c5 60   1f
c550 : ee 26 d0 20 81 c1 20 90   51
c558 : c5 60 ce 25 d0 20 81 c1   3d
c560 : 20 90 c5 60 ce 26 d0 20   e7
c568 : 81 c1 20 90 c5 60 a9 0f   08
c570 : cd 54 c0 d0 03 20 50 c3   ab
c578 : 20 50 c3 20 8c c1 60 a9   61
c580 : 0f cd 54 c0 d0 03 20 70   29
c588 : c3 20 70 c3 20 8c c1 60   1e
c590 : 20 8c c1 ad 27 d0 ae 25   1a
c598 : d0 ac 26 d0 8d 45 db 8d   ef
c5a0 : 46 db 8e 6d db 8e 6e db   c9
c5a8 : 8c 95 db 8c 96 db 60 18   81
c5b0 : a2 17 a0 00 20 f0 ff a9   e3
c5b8 : 01 a2 08 a0 0f 20 ba ff   fd
c5c0 : a9 00 20 bd ff 20 c0 ff   2d
c5c8 : a2 01 20 c6 ff 20 cf ff   0c
c5d0 : a8 20 cf ff a2 0e 8e 86   5e
c5d8 : 02 aa 98 20 d2 ff 8a 20   f1
c5e0 : d2 ff 20 cf ff 20 d2 ff   00
c5e8 : a9 02 8d 86 02 20 cf ff   27
c5f0 : c9 2c f0 06 20 d2 ff 4c   fd
c5f8 : ed c5 a2 0e 8e 86 02 20   98
c600 : d2 ff 20 cf ff 20 d2 ff   20
c608 : 24 90 50 f6 20 cc ff a9   23
c610 : 01 20 c3 ff 60 a9 00 8d   81
c618 : 15 d0 20 af c5 a9 8f 8d   96
c620 : 15 d0 60 ae 00 dc 8a 29   ef
c628 : 01 f0 25 8a 29 02 f0 13   c9
c630 : 8a 29 04 f0 34 8a 29 08   ba
c638 : f0 23 8a 29 10 f0 36 4c   7c
c640 : 83 c6 0e ae 42 c6 e8 fe   7c
c648 : 00 d0 20 76 c6 4c 23 c6   70
c650 : ae 42 c6 e8 de 00 d0 20   5f
c658 : 76 c6 4c 23 c6 ae 42 c6   21
c660 : fe 00 d0 20 76 c6 4c 23   ac
c668 : c6 ae 42 c6 de 00 d0 20   60
c670 : 76 c6 4c 23 c6 60 a0 00   b3
c678 : a2 00 e8 d0 fd c8 c0 07   a6
c680 : d0 f6 60 a6 cb e0 40 d0   1f
c688 : 03 4c 23 c6 e0 28 d0 08   f6
c690 : a9 07 8d 7f c6 4c 23 c6   f9
c698 : e0 2b d0 05 a9 20 8d 7f   b3
c6a0 : c6 4c 23 c6 ee 20 d0 ee   3f
c6a8 : 21 d0 20 76 c6 60 ea ea   f9
c6b0 : ea ea ea ea ea ea ea ea   af
c6b8 : ea ea ea ea ea ea ea ea   b7
c6c0 : ea ea ea ea ea ea ea ea   bf
c6c8 : ea ea ea ea ea ea ea ea   c7
c6d0 : ea ea ea ea ea ea ea ea   cf
c6d8 : ea ea ea ea ea ea ea ea   d7
c6e0 : ea ea ea ea ea ea ea ea   df
c6e8 : ea ea ea ea ea ea ea ea   e7
c6f0 : ea ea ea ea ea ea ea ea   ef
c6f8 : ea ea ea ea ea ea ea ea   f7
```

Listing 2. Maschinen-Routinen zu »Graphic Art«. Bitte beachten Sie die Eingabehinweise auf Seite 7

3D-Grafik für Schachspiele

Wollen Sie Ihrem Schachprogramm zu einer räumlichen Darstellung verhelfen oder wichtige Schachpartien speichern? Hier finden Sie wertvolle Routinen dafür.

Haben Sie sich schon häufig über die zweidimensionale Darstellung bei Schachspielen geärgert? Das Programm »Schachgrafik« (Listing) beendet diese Misere. Sie können damit ein perspektivisches Schachbrett erzeugen, wie es im Bild zu sehen ist.

Das Programm ist vollständig in Maschinensprache geschrieben und belegt den Speicherbereich $C000 bis $C9BE. Der Basic-Speicher bleibt auch bei eingeschalteter Grafik frei. Nachdem Sie das Programm mit dem MSE eingegeben haben, stehen Ihnen zwölf Routinen zur Verfügung, die mit dem SYS-Befehl aufgerufen werden können. Schachprogrammen wie »Schachmeister« (Ausgabe 11/84) oder »Schach dem C 64« (Ausgabe 8/85) verhelfen Sie mit diesem Programm zu einer hervorragenden Grafik.

So, nun zu den einzelnen Routinen, die Ihnen nach der Eingabe von »Schachgrafik« zur Verfügung stehen.

SYS 50000

Hiermit schalten Sie die Grafik ein. Weder die Bitmap wird gelöscht, noch werden die Farben der Grafik gesetzt. Die Bitmap liegt bei $E000, der Farbspeicher bei $CC00.

SYS 50003,FLAG

Schaltet die Grafik aus. Bei FLAG=0 geht der Computer in den Großschrift-Grafikmodus, bei FLAG=1 in den Kleinschrift-Großschrift-Modus.

SYS 50006

Löschen der Bitmap

SYS 50009,Byte

Setzen der Grafikfarben. Werte zwischen 0 und 255 können eingegeben werden. Das obere Nibble des Bytes bestimmt die Zeichen-, das untere die Hintergrundfarbe.

SYS 50012,FLAG

Zeichnen eines leeren Brettes. Bei FLAG=0 ist die untere linke Ecke in der Zeichen-, bei FLAG=1 in der Hintergrundfarbe gezeichnet.

SYS 50015,FIG,FELD

Dieser Befehl setzt die Figur mit dem Code FIG auf das Feld mit der Nummer FELD. Die Figurencodes:

Farbe 1		Farbe 2	Bedeutung
0	oder	128	:Leeres Feld
1	oder	129	:Bauer
2	oder	130	:Pferd
3	oder	131	:Läufer
4	oder	132	:Turm
5	oder	133	:Dame
6	oder	134	:König

Alle weiteren Codes sind nicht erlaubt. Die Felder sind von 0 bis 63 durchnumeriert. Dabei liegt Feld 0 links oben.

SYS 50018,FELD

Zeichnet nur das Feld mit der Nummer FELD neu.

SYS 50021,FELDA,FELDB

Setzt die Figur auf FELDA nach FELDB und zeichnet beide Felder neu.

SYS 50024

Zeichnet alle Felder neu. Nach dem Laden der Schachgrafik ist das Schachbrett in der Anfangsstellung besetzt.

SYS 50027

Rotiert das Brett einmal gegen den Uhrzeigersinn. Hierbei werden auch synchron die Nummern der Felder geändert. Nach einmaligem Drehen befindet sich das Feld 0 links unten, nach zweimaligem Drehen rechts unten etc.

SYS 50030

Nach Ausführung dieses Befehls sind die Feldnummern wieder so, als sei das Brett bisher nicht gedreht worden. Feld Nr. 0 ist wieder links oben.

SYS 50033,TEXT,ZEILE,SPALTE

Gibt den TEXT an der durch ZEILE und SPALTE bestimmten Position in der Bitmap aus. Alle Zeichen erscheinen im Großschrift-Grafikmodus. Steuerzeichen werden als Leerzeichen interpretiert.

Es muß gelten: 0 <=ZEILE <=24 und 0 <=SPALTE <=39.

Bei allen Befehlen können alle zu übergebenden Werte in beliebigen Formen, zum Beispiel als Variablen, Formeln oder ähnliches auftauchen.

Leider schreibt bei einem RUN/STOP-RESTORE eine ROM-Routine (bei $FD15) 32 Bytes in die Bitmap, die man aber mit SYS 50033," ",23,14 löschen kann. Trotzdem sollte man RUN/STOP-RESTORE vermeiden oder sogar softwaremäßig unterbinden.

Liste aller Datensätze und Routinen

$C000-$C29F	AND- und OR-Masken (je 48 Bytes) von Unterteil, Bauer, Pferd, Läufer, Turm, Dame und König.
$C2A0-$C2FF	3*4-Grafikausschnitt Brett (als Hintergrund).
$C300-$C30F	2 quadratische Grafikausschnitte, die in einer Routine benötigt werden.
$C310-$C34F	Speicher des Feldes
$C350-$C373	Sprungtabelle zu den Hauptroutinen.
$C374-$C4FE	Die Hauptroutinen aller Befehle, Beginn der einzelnen Routinen siehe Sprungtabelle.
$C4FF-$C607	Zeichnet Text in Bitmap. Leicht korrigiert übernommen von Heimo Ponnath, 64'er, 8/85.
$C608-$C62C	4 Routinen, die prüfen, ob ein Feld am Rand des Brettes liegt.
$C62D-$C696	Hier werden alle Teile des Brettes, die in keinem 3*4-Feldausschnitt enthalten sind, gezeichnet.
$C697-$C752	überträgt 3*4-Brettausschnitt (der Feldnummer entsprechend modifiziert) in den Puffer.
$C753-$C76E	Rotiert einmal das gesamte Feld.
$C76F-$C7B1	Errechnet Adresse eines Grafikausschnittes in der Bitmap abhängig von der Feldnummer.

$C7B2-$C7E2	Überträgt Pufferinhalt in die Bitmap.
$C7E3-$C80F	Überträgt Figur eines Feldes in den Puffer.
$C810-$C847	Überträgt unteren rechten Ausschnitt der Figur, die im Grafikausschnitt oben links auftaucht, in den Puffer.
$C848-$C87A	Überträgt unteren linken Ausschnitt der Figur, die im Grafikausschnitt oben rechts auftaucht, in den Puffer.
$C87B-$C8AD	Überträgt oberen rechten Ausschnitt der Figur, die im Grafikausschnitt unten links auftaucht, in den Puffer.
$C8AE-$C8E5	Überträgt oberen linken Ausschnitt der Figur, die im Grafikausschnitt unten rechts auftaucht, in den Puffer.
$C8E6-$C8F9	Rotiert eine im X-Register stehende Feldnummer ein- oder mehrmals gegen den Uhrzeigersinn.
$C8FA-$C914	Verknüpft zwei Bytes aus zwei AND- bzw. OR-Masken mit dem Pufferinhalt.
$C915-$C923	Adressen der Unterteilmasken.
$C924-$C949	Ermittelt Adressen der Masken beliebiger Figuren und schreibt sie in zwei Zeiger.
$C94A-$C95C	Rotiert eine im X-Register stehende Feldnummer einmal gegen den Uhrzeigersinn.
$C95D-$C95E	Zwei Flags.
$C95F-$C9BE	Puffer.

Auf der Leserservice-Diskette finden Sie zusätzlich ein Demonstrationsprogramm, das Ihnen alle Möglichkeiten von »Schachgrafik« aufzeigt. (Frithof Dau/kn)



Bild: Hardcopy zu »Schachgrafik«

```
programm : schachgrafik     c000 c990
-------------------------------------
c000 : ff ff ff fe f8 f0 e0 e0  3b
c008 : 81 81 81 00 00 00 00 00  aa
c010 : ff ff ff 7f 1f 0f 07 07  93
c018 : e0 e0 f0 f8 fe ff ff ff  b3
c020 : 00 00 00 00 00 81 ff ff  2d
c028 : 07 07 0f 1f 7f ff ff ff  53
c030 : 00 00 00 01 06 08 14 12  66
c038 : 42 42 42 bd 00 00 00 00  e4
c040 : 00 00 00 80 60 10 28 48  08
c048 : 11 10 08 06 01 00 00 00  34
c050 : 81 7e 00 00 81 7e 00 00  1d
c058 : 88 08 10 60 80 00 00 00  fd
c060 : ff ff ff ff ff ff ff ff  5f
c068 : ff ff ff ff ff ff ff e7  36
c070 : ff ff ff ff ff ff ff ff  6f
c078 : ff ff ff fe fe fe ff ff  3e
c080 : c3 81 00 00 00 00 00 c3  8c
c088 : ff ff ff 7f 7f 7f ff ff  6b
c090 : 00 00 00 00 00 00 00 00  91
c098 : 00 00 00 00 00 00 00 18  c9
c0a0 : 00 00 00 00 00 00 00 00  a1
c0a8 : 00 00 00 01 01 01 00 00  e1
c0b0 : 24 5a 81 00 00 00 99 42  4d
c0b8 : 00 00 00 80 80 80 00 00  d5
c0c0 : ff ff ff ff ff ff ff fe  bd
c0c8 : ff ff ff ff c3 81 00 00  0f
c0d0 : ff ff ff ff ff ff ff 7f  ce
c0d8 : fc f8 f0 f0 f0 f8 ff ff  81
c0e0 : 00 00 00 00 40 80 80 81  ee
c0e8 : 7f 7f 7f 7f 7f 7f ff ff  eb
c0f0 : 00 00 00 00 00 00 00 01  f3
c0f8 : 00 00 00 00 3c 42 81 30  35
c100 : 00 00 00 00 00 00 00 80  02
c108 : 02 04 08 08 08 07 00 00  c8
c110 : 02 02 02 60 a0 40 59 42  96
c118 : 80 80 80 80 80 80 00 00  15
c120 : ff ff ff ff ff ff ff fe  1d
c128 : ff ff ff ff c3 81 80 c0  f2
c130 : ff ff ff ff ff ff ff 7f  2e
c138 : fe fe fe fe fe fe ff ff  3d
c140 : 60 20 00 00 00 00 00 81  b4
c148 : 7f 7f 7f 7f 7f 7f ff ff  4b
c150 : 00 00 00 00 00 00 00 01  53
c158 : 00 00 00 00 3c 42 41 20  74
c160 : 00 00 00 00 00 00 00 80  62
c168 : 01 01 01 01 01 01 00 00  62
c170 : 90 50 30 00 00 00 99 42  1f
c178 : 80 80 80 80 80 80 00 00  75
c180 : ff ff ff ff ff f8 f8 f8  1c
c188 : ff ff ff ff ff 42 42 42  26
c190 : ff ff ff ff ff 1f 1f 1f  42
c198 : f8 f8 f8 f8 fc fe ff ff  31
c1a0 : 00 00 00 00 00 00 00 81  a4
c1a8 : 1f 1f 1f 1f 3f 7f ff ff  f3
c1b0 : 00 00 00 00 00 07 04 04  01
c1b8 : 00 00 00 00 00 bd a5 a5  88
c1c0 : 00 00 00 00 00 e0 20 20  88
c1c8 : 04 04 04 04 02 01 00 00  78
c1d0 : e7 00 00 3c 00 00 99 42  2a
c1d8 : 20 20 20 20 40 80 00 00  1d
c1e0 : ff ff ff ff fe fc fc fe  a8
c1e8 : ff ff 81 00 00 00 00 00  47
c1f0 : ff ff ff ff 7f 3f 3f 7f  dd
c1f8 : ff ff ff ff fe fe ff ff  de
c200 : 00 81 81 00 00 00 00 81  24
c208 : ff ff ff ff 7f 7f ff ff  fb
c210 : 00 00 00 00 01 02 02 01  3b
c218 : 00 00 7e 81 00 [illegible] 00 00  ca
c220 : 00 00 00 00 80 40 40 80  2d
c228 : 00 00 00 00 01 01 00 00  41
c230 : 81 42 42 81 00 00 99 42  7e
c238 : 00 00 00 00 80 80 00 00  45
c240 : ff ff ff ff ff ff ff ff  3f
c248 : ff ff c3 c3 00 00 00 00  b0
c250 : ff ff ff ff ff ff ff ff  4f
c258 : ff ff ff fe fe fe ff ff  1e
c260 : c3 81 00 00 00 00 00 c3  6c
c268 : ff ff ff 7f 7f 7f ff ff  4b
c270 : 00 00 00 00 00 00 00 00  71
c278 : 00 00 3c 24 e7 81 81 e7  6c
c280 : 00 00 00 00 00 00 00 00  81
c288 : 00 00 00 01 01 01 00 00  c1
c290 : 24 5a 81 00 00 00 99 42  2d
c298 : 00 00 00 80 80 80 00 00  b5
c2a0 : ff ff ff ff ff ff ff ff  9f
c2a8 : ff ff ff ff ff ff ff ff  a7
c2b0 : ff fe fc f8 f0 e0 c0 80  a7
c2b8 : ff ff ff ff ff ff ff ff  b7
c2c0 : ff fe fc f8 f0 e0 c0 80  b7
c2c8 : 00 00 00 00 00 00 00 00  c9
c2d0 : 00 01 03 07 0f 1f 3f 7f  d9
c2d8 : ff ff ff ff ff ff ff ff  d7
c2e0 : ff ff ff ff ff ff ff ff  df
c2e8 : ff ff ff ff ff ff ff ff  e7
c2f0 : ff ff ff ff ff ff ff ff  ef
c2f8 : ff fe fc f8 f0 e0 c0 80  ef
c300 : ff fe fc f8 f0 e0 c0 80  f7
c308 : 01 02 04 08 10 20 40 80  11
c310 : 84 82 83 85 86 83 82 84  fe
c318 : 81 81 81 81 81 81 81 81  17
c320 : 00 00 00 00 00 00 00 00  21
c328 : 00 00 00 00 00 00 00 00  29
c330 : 00 00 00 00 00 00 00 00  31
c338 : 00 00 00 00 00 00 00 00  39
c340 : 01 01 01 01 01 01 01 01  40
c348 : 04 02 03 05 06 03 02 04  37
c350 : 4c 74 c3 4c b6 c4 4c 8f  33
c358 : c3 4c c5 c3 4c e3 c3 4c  b7
c360 : 6d c4 4c 8e c4 4c 08 c4  6d
c368 : 4c d8 c4 4c a8 c3 4c f9  a9
c370 : c4 4c 02 c5 ad 00 dd 29  38
c378 : fc 8d 00 dd ad 11 d0 09  af
c380 : 20 8d 11 d0 ad 18 d0 29  f6
c388 : 0f 09 38 8d 18 d0 60 a9  b8
c390 : 00 85 5b a9 e0 85 5c a2  50
c398 : 1f a0 00 a9 00 91 5b c8  c8
c3a0 : d0 fb e6 5c ca 10 f6 60  7d
c3a8 : ee 5e c9 ad 5e c9 29 03  cc
c3b0 : 8d 5e c9 20 53 c7 ad 5d  c8
c3b8 : c9 49 01 8d 5d c9 20 f3  a4
c3c0 : c3 20 d8 c4 60 20 fd ae  be
c3c8 : 20 9e b7 a9 cc 85 5c a9  18
c3d0 : 00 85 5b a0 00 8a a2 03  63
c3d8 : 91 5b c8 d0 fb e6 5c ca  61
c3e0 : 10 f6 60 20 fd ae 20 9e  9a
c3e8 : b7 e0 02 b0 7d 8a 29 01  79
c3f0 : 8d 5d c9 a9 3f 85 02 20  3c
c3f8 : 97 c6 20 6f c7 20 b2 c7  c0
c400 : c6 02 10 f3 20 2d c6 60  91
c408 : 20 fd ae 20 9e b7 e0 40  82
c410 : b0 58 20 e6 c8 8a 85 02  cc
c418 : 20 fd ae 20 9e b7 e0 40  92
c420 : b0 48 20 e6 c8 8a 48 a4  24
c428 : 02 b9 10 c3 48 a9 00 99  88
c430 : 10 c3 20 97 c6 20 10 c8  5c
c438 : 20 48 c8 20 7b c8 20 ae  8e
c440 : c8 20 6f c7 20 b2 c7 68  75
c448 : aa 68 a8 85 02 8a 99 10  fc
c450 : c3 20 97 c6 20 10 c8 20  c8
c458 : 48 c8 20 e3 c7 20 7b c8  86
c460 : 20 ae c8 20 6f c7 20 b2  29
c468 : c7 60 4c 48 b2 20 fd ae  fd
c470 : 20 9e b7 8a 29 7f c9 07  e2
c478 : b0 f0 8a 48 20 fd ae 20  39
c480 : 9e b7 e0 40 b0 e4 20 e6  ba
c488 : c8 68 9d 10 c3 60 20 fd  a9
c490 : ae 20 9e b7 e0 40 b0 d2  65
c498 : 20 e6 c8 86 02 20 97 c6  3b
c4a0 : 20 10 c8 20 48 c8 20 e3  12
c4a8 : c7 20 7b c8 20 ae c8 20  52
c4b0 : 6f c7 20 b2 c7 60 20 fd  5d
c4b8 : ae 20 9e b7 e0 02 b0 aa  4b
c4c0 : ad 00 dd 09 03 8d 00 dd  5e
c4c8 : ad 11 d0 29 df 8d 11 d0  a7
c4d0 : 8a 0a 09 14 8d 18 d0 60  c2
c4d8 : a9 3f 85 02 20 97 c6 20  dd
c4e0 : 10 c8 20 48 c8 20 e3 c7  12
c4e8 : 20 7b c8 20 ae c8 20 6f  8c
c4f0 : c7 20 b2 c7 c6 02 10 e4  f3
c4f8 : 60 a9 00 8d 5e c9 60 4c  2d
c500 : 48 b2 20 fd ae 20 9e ad  2b
c508 : a0 00 b1 64 85 24 c8 b1  a1
c510 : 64 85 04 c8 b1 64 85 05  af
c518 : a5 64 a4 65 20 db b6 20  c1
c520 : fd ae 20 9e b7 e0 19 b0  98
c528 : d6 86 5b a9 40 85 59 a9  36
c530 : 01 85 5a 20 84 c5 20 fd  81
c538 : ae 20 9e b7 e0 28 b0 bf  26
c540 : 8a 18 2a 2a 2a 85 25 a9  5d
c548 : 00 2a 85 26 18 a5 57 65  5a
c550 : 25 85 25 a5 58 65 26 85  8a
c558 : 26 18 a5 25 69 00 85 25  8f
c560 : a5 26 69 e0 85 26 a0 00  9b
c568 : a5 24 f0 17 b1 04 aa 98  55
c570 : 48 8a 20 a2 c5 20 c6 c5  5e
c578 : f0 08 68 a8 c8 c4 24 30  3f
c580 : eb 60 68 60 a9 00 85 57  21
```

Listing: MSE-Listing des Programms »Schachgrafik«

```
c588 : 85 58 a6 5b ca e0 ff f0  e4
c590 : 10 18 a5 59 65 57 85 57  17
c598 : a5 5a 65 58 85 58 4c 8c  34
c5a0 : c5 60 10 03 4c b6 c5 c9  1f
c5a8 : 20 90 18 c9 60 90 04 29  3d
c5b0 : df d0 02 29 3f 60 29 7f  38
c5b8 : c9 7f d0 02 a9 5e c9 20  aa
c5c0 : 90 01 60 a9 20 60 a2 00  ae
c5c8 : 86 27 86 29 a2 d0 86 28  c4
c5d0 : 18 2a 26 29 2a 26 29 2a  79
c5d8 : 26 29 18 65 27 85 27 a5  cc
c5e0 : 28 65 29 85 28 a0 07 a5  a5
c5e8 : 01 48 29 fb 78 85 01 b1  f2
c5f0 : 27 91 25 88 10 f9 68 85  b8
c5f8 : 01 58 18 a5 25 69 08 85  a9
c600 : 25 a5 26 69 00 85 26 60  34
c608 : a5 02 c9 08 90 02 18 60  5c
c610 : 38 60 a5 02 c9 38 60 a5  4d
c618 : 02 29 07 f0 02 18 60 38  62
c620 : 60 a5 02 29 07 c9 07 f0  b5
c628 : 02 18 60 38 60 a9 b8 85  97
c630 : 5b a9 e8 85 5c ae 5d c9  8f
c638 : a9 08 85 5e a0 07 e0 01  da
c640 : f0 0b b9 b0 c2 91 5b 88  71
c648 : 10 f8 4c 55 c6 b9 08 c3  74
c650 : 91 5b 88 10 f8 8a 49 01  be
c658 : aa a5 5b 18 69 70 85 5b  96
c660 : a5 5c 69 02 85 5c c6 5e  e1
c668 : d0 d2 a9 40 85 5b a9 fa  e4
c670 : 85 5c ad 5d c9 f0 0d a0  d4
c678 : 07 b9 08 c3 91 5b 88 10  0d
c680 : f8 4c 8e c6 a0 07 b9 d1  e7
c688 : c2 91 5b 88 10 f8 a9 ff  6a
c690 : 8d 47 fa 8d 7f e7 60 a5  35
c698 : 02 29 09 f0 0a c9 09 d0  44
c6a0 : 04 a9 00 f0 02 a9 01 4d  a3
c6a8 : 5d c9 29 01 aa a0 5f b9  f5
c6b0 : a0 c2 e0 01 f0 02 49 ff  4e
c6b8 : 99 5f c9 88 10 f1 20 17  c3
c6c0 : c6 90 39 a0 07 a9 00 99  22
c6c8 : 5f c9 99 67 c9 99 77 c9  3a
c6d0 : 88 10 f4 a0 07 b9 00 c3  77
c6d8 : 49 ff 48 39 6f c9 19 08  14
c6e0 : c3 99 6f c9 68 48 39 7f  32
c6e8 : c9 19 08 c3 99 7f c9 68  46
c6f0 : 39 8f c9 19 08 c3 99 8f  aa
c6f8 : c9 88 10 d9 20 08 c6 90  c3
c700 : 15 a0 2f a9 00 99 5f c9  44
c708 : 88 10 fa a9 ff 8d 7e c9  86
c710 : 8d 86 c9 8d 8e c9 20 21  fe
c718 : c6 90 11 a0 07 b9 b7 c9  2f
c720 : 39 00 c3 19 08 c3 99 b7  e2
c728 : c9 88 10 f1 20 12 c6 90  46
c730 : 0b a9 ff 8d ae c9 8d b6  9e
c738 : c9 8d be c9 a5 02 c9 00  42
c740 : d0 05 a9 00 8d 7e c9 a5  3c
c748 : 02 c9 3f d0 05 a9 00 8d  d1
c750 : be c9 60 a2 3f bd 10 c3  09
c758 : 9d 5f c9 ca 10 f7 a2 3f  3a
c760 : bd 5f c9 48 20 4a c9 68  95
c768 : 99 10 c3 ca 10 f2 60 a9  c1
c770 : 00 85 5c a5 02 29 07 85  8f
c778 : 5b 0a 18 65 5b 85 5b a5  26
c780 : 02 4a 4a 4a 48 a0 4d 68  13
c788 : 48 18 65 5b 85 5b a5 5c  24
c790 : 69 00 85 5c 88 10 f0 06  bf
c798 : 5b 26 5c 06 5b 26 5c 06  43
c7a0 : 5b 26 5c 68 18 a5 5b 69  21
c7a8 : 78 85 5b a5 5c 69 e5 85  22
c7b0 : 5c 60 a9 5f a2 c9 85 59  d4
c7b8 : 86 5a a2 03 a0 17 b1 59  b1
c7c0 : 91 5b 88 10 f9 18 a5 59  cd
c7c8 : 69 18 85 59 a5 5a 69 00  9d
c7d0 : 85 5a 18 a5 5b 69 40 85  4a
c7d8 : 5b a5 5c 69 01 85 5c ca  8d
c7e0 : 10 da 60 a4 02 b9 10 c3  c0
c7e8 : 85 5d 29 7f f0 21 20 15  19
c7f0 : c9 a9 8f 85 59 a9 c9 85  37
c7f8 : 5a 20 07 c8 20 24 c9 a9  db
c800 : 5f 85 59 a9 c9 85 5a a0  21
c808 : 2f 20 fa c8 88 10 fa 60  d5
c810 : 20 08 c6 b0 2d 20 17 c6  ba
c818 : b0 28 a5 02 38 e9 09 a8  ce
c820 : b9 10 c3 85 5d 29 7f f0  82
c828 : 19 20 15 c9 a9 57 85 59  ee
c830 : a9 c9 85 5a a0 2f 20 fa  64
c838 : c8 88 c0 1f f0 05 c0 07  a1
c840 : d0 f4 60 a0 17 4c 36 c8  f5
c848 : 20 08 c6 b0 28 a5 02 38  5c
c850 : e9 08 a8 b9 10 c3 85 5d  8f
c858 : 29 7f f0 19 20 15 c9 a9  c5
c860 : 6f 85 59 a9 c9 85 5a a0  91
c868 : 1f 20 fa c8 88 c0 17 f0  3c
c870 : 05 c0 ff d0 f4 60 a0 07  d2
c878 : 4c 69 c8 20 12 c6 b0 28  19
c880 : a5 02 18 69 08 a8 b9 10  26
c888 : c3 85 5d 29 7f f0 19 20  ae
c890 : 24 c9 a9 7f 85 59 a9 c9  50
c898 : 85 5a a0 2f 20 fa c8 88  66
c8a0 : c0 27 f0 05 c0 0f d0 f4  82
c8a8 : 60 a0 17 4c 9c c8 20 12  5c
c8b0 : c6 b0 2d 20 21 c6 b0 28  79
c8b8 : a5 02 18 69 09 a8 b9 10  6e
c8c0 : c3 85 5d 29 7f f0 19 20  e6
c8c8 : 24 c9 a9 97 85 59 a9 c9  8b
c8d0 : 85 5a a0 27 20 fa c8 88  9d
c8d8 : c0 17 f0 05 c0 ff d0 f4  3a
c8e0 : 60 a0 0f 4c d4 c8 48 ad  ee
c8e8 : 5e c9 85 5e c6 5e 30 08  88
c8f0 : 20 4a c9 98 aa 4c ec c8  0d
c8f8 : 68 60 b1 59 31 5b 91 59  0f
c900 : b1 57 a6 5d 10 0a 49 ff  29
c908 : 85 5e b1 5b 49 ff 25 5e  7a
c910 : 11 59 91 59 60 a9 c0 85  bf
c918 : 5c 85 58 a9 00 85 5b a9  6f
c920 : 30 85 57 60 20 15 c9 a5  12
c928 : 5d 29 7f aa 18 a5 5b 69  3e
c930 : 60 85 5b a5 5c 69 00 85  fa
c938 : 5c ca d0 f0 a5 5b 18 69  b4
c940 : 30 85 57 a5 5c 69 00 85  d9
c948 : 58 60 8a 48 29 07 49 07  7a
c950 : 0a 0a 0a 85 5f 68 4a 4a  8a
c958 : 4a 05 5f a8 60 00 00 20  58
c960 : 41 58 4f 4e 2d 53 4f 46  a2
c968 : 54 57 41 52 45 20 3c 43  cf
c970 : 3e 20 31 39 38 35 20 20  20
c978 : 46 52 49 54 48 4a 4f 46  65
c980 : 20 44 41 55 20 20 20 20  81
c988 : 20 20 20 20 20 20 20 20  88
```

Listing. »Schachgrafik« (Schluß)

64ER ONLINE

Super Hardcopy für den MPS 802

Ein fantastisches Hardcopy-Programm für den MPS 802. Es druckt genau das aus, was im Moment auf dem Bildschirm zu sehen ist – egal, ob es der Text-Bildschirm oder eine HiRes-Grafik ist.

Obwohl der MPS 802 in der Grundausstattung nicht grafikfähig ist, gibt es seit kurzer Zeit schon Hardcopy-Programme, die einen Grafikausdruck mit diesem Drucker ermöglichen. Viele jedoch sind entweder nur sehr beschränkt anwendbar oder aber teilweise fehlerhaft. Dieses Programm bietet Ihnen folgende Vorzüge:

1. Es wird wirklich das aufs Papier gebracht, was sich zur Zeit des Drucks am Bildschirm befindet. Es ist hierbei egal, ob es sich um eine HiRes-Grafik handelt oder um den normalen Textbildschirm.

2. Der Beginn des Bildschirmspeichers beziehungsweise des HiRes-Bit-Map wird in jedem Fall vom Programm selbst ermittelt. (Im Speicher befindliche Bilder müssen zuerst auf den Bildschirm gebracht werden.)

3. Im Falle einer LowRes-Hardcopy kann durch einfaches Drücken der SHIFT- und COMMODORE-Taste gleichzeitig zwischen Großschrift/Grafik-Modus und Klein/Großschrift-Modus gewählt werden. (Natürlich vor dem Aufruf der Routine.)

4. Alle möglichen Parameter gelten sowohl für eine Hardcopy des LowRes- als auch des HiRes-Bildschirms.

5. Einen besonderen Leckerbissen bietet das Programm im Erstellen von 4mal so großen HiRes-Hardcopys, ohne daß sich die Proportion im Vergleich zum Bildschirm ändert. Eine normale Hardcopy dieser Art benötigt zirka 15 bis 45 Minuten.

6. Ein weiterer Vorteil besteht im Verbinden mehrerer Hardcopybilder, was sowohl längs mit Hilfe des Tabulators geschieht als auch untereinander, da nach beendetem Ausdruck des ersten Bildes kein Zeilenvorschub getätigt wird.

7. Die Invertierung des HiRes-Bildes kann oft von Vorteil sein (siehe Bild 1 und 2). Bei Invertierung des LowRes-Bildes ist jedoch Vorsicht geboten, da der Druckknopf und das Farbband hierbei sehr in Mitleidenschaft gezogen werden.

8. Durch ein spezielles Unterprogramm kann auch die HiRes-Bit-Map unter dem ROM gelesen werden, das heißt, daß die HiRes-Bit-Map in jedem Bereich außer zwischen $C000 und $DFFF liegen darf (also auch für die bekannten Befehlserweiterungen wie Simons Basic geeignet!), da sich in diesem Bereich das Programm und der VIC-Steuerchip befinden.

64er-online.de

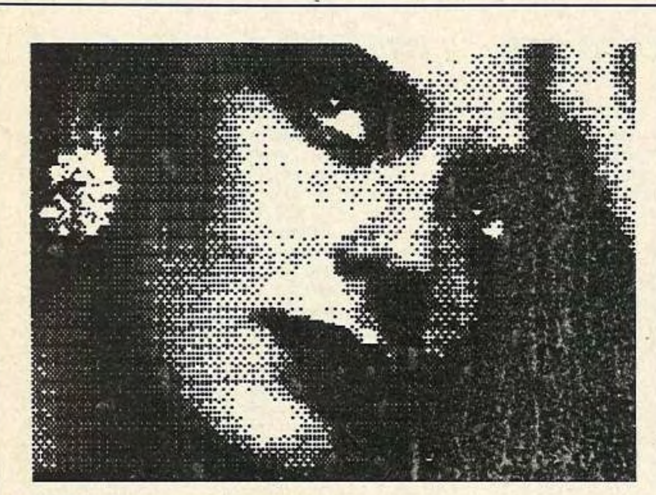


Bild 1. Hardcopy: invertiert...

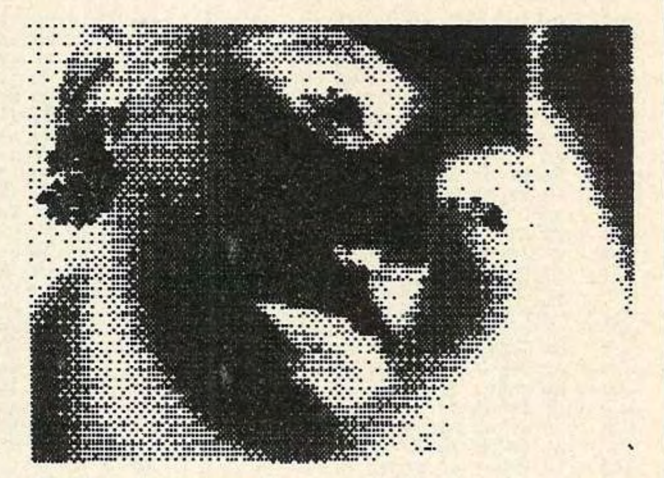


Bild 2. ...oder normal

Programmbeschreibung

1. Programmierung: Das Programm (Listing 1) ist in 6510 Maschinensprache geschrieben und auf Grund seiner Vielfältigkeit stark in Unterprogramme gegliedert. Im Prinzip besteht es aber aus drei großen Hauptabschnitten, nämlich der normalen Low- beziehungsweise HiRes-Hardcopy sowie der vierfach vergrößerten HiRes-Hardcopy, der wegen ihrer außergewöhnlichen Ausführung ein extra Programmabschnitt zugewiesen worden ist. Einzelheiten hierzu können aus dem kommentierten Assemblerlisting (Listing 2) entnommen werden, das wir Ihnen nicht vorenthalten möchten.

2. Ausführungsdaten: Das Programm läuft erwartungsgemäß sehr schnell ab und benötigt für eine einfache HiRes-Hardcopy je nach Größe zwischen einer und fünf Minuten. Eine LowRes-Hardcopy dauert maximal eine Minute. Hingegen braucht der Drucker für eine vierfache HiRes-Hardcopy zwischen 5 und 45 Minuten, weil der Weg, den der Druckkopf zurücklegt, sich vervielfacht.

3. Speicheraufteilung: Das als MSE-Listing abgedruckte Programm startet bei $C000 und endet bei $C332. Sollte dieser Bereich bei Ihnen belegt sein, so ändern Sie die Startadresse des Assemblerprogramms und assemblieren es neu.

4. Syntax: SYS 49152, Größe, Invertierung, Tabulator
Parameter: a) Größe: 1 = normal (9,5 x 6,2 cm - 320 x 200 einfache Nadelpunkte)
2 = doppelt (19 x 12,4 cm - 320 x 200 vierfache Nadelpunkte)
b) Invertierung: 0 = ohne Invertierung
1 = mit Invertierung
c) Tabulator: von 0 bis 40 wenn Größe = 1,
fällt weg, wenn Größe = 2

5. Verwendung: Das auf Diskette (oder Kassette) durch den MSE abgespeicherte Programm mit LOAD"HC MPS-802",8,1 laden, danach NEW zur Rückstellung der Zeiger eingeben. Sobald Sie das Programm gestartet haben, können Sie es jederzeit mit einem einfachen Druck auf die STOP-Taste unterbrechen, alle geöffneten Kanäle werden ordnungsgemäß wieder geschlossen.

Nun ist der letzte Zweifel über die Grafikfähigkeit des MPS-802 beseitigt. Viel Vergnügen mit Ihrem neuen Hardcopy-Programm! (R. Sucher/og)

```
programm : hc mps-802       c000 c332
--------------------------------------
c000 : 20 28 c0 e0 03 b0 28 86  e4
c008 : fb 20 28 c0 e0 02 b0 1f  55
c010 : 86 fc a0 00 84 fd a4 fb  ff
c018 : c0 02 f0 16 20 28 c0 e0  e0
c020 : 29 b0 0c 86 fd 4c 32 c0  02
c028 : 18 20 fd ae 4c 9e b7 4c  d7
c030 : 48 b2 ad 02 dd 09 03 8d  ca
c038 : 02 dd 20 e7 ff a9 04 a2  d0
c040 : 04 a0 00 20 94 c0 a9 05  98
c048 : a2 04 a0 05 20 94 c0 a9  b2
c050 : 06 a2 04 a0 06 20 94 c0  f2
c058 : a2 06 20 c9 ff a9 15 20  20
c060 : d2 ff a9 0d 20 d2 ff 20  17
c068 : cc ff a9 00 85 fe 85 72  e9
c070 : ad 00 dd 49 ff 29 03 18  43
c078 : 6a 6a 6a 85 ff ad 11 d0  b6
c080 : 29 20 c9 20 d0 19 ad 18  ec
c088 : d0 29 08 0a 0a 05 ff 85  04
c090 : ff 4c a2 c0 20 ba ff a9  a1
c098 : 00 20 bd ff 4c c0 ff 4c  7b
c0a0 : 1e c1 a5 fb c9 02 f0 24  40
c0a8 : a9 00 85 71 20 e1 ff f0  d4
c0b0 : 18 20 92 c1 f0 06 20 eb  4d
c0b8 : c1 20 05 c2 20 27 c2 d0  0b
c0c0 : eb 20 39 c2 20 46 c2 d0  43
c0c8 : df 4c 6b c1 a5 ff 48 a5  a7
c0d0 : fe 48 a9 00 85 71 20 e1  85
c0d8 : ff f0 40 20 92 c1 f0 09  70
c0e0 : 20 71 c2 20 eb c1 20 4d  55
c0e8 : c2 20 27 c2 d0 e8 20 39  24
c0f0 : c2 68 85 fe 68 85 ff a9  2d
c0f8 : 00 85 71 20 e1 ff f0 1b  33
c100 : 20 92 c1 f0 09 20 9c c2  81
c108 : 20 eb c1 20 4d c2 20 27  4c
c110 : c2 d0 e8 20 39 c2 20 46  2f
c118 : c2 d0 b1 4c 6b c1 ad 18  e4
c120 : d0 29 f0 4a 4a 05 ff 85  e2
c128 : ff a2 06 20 c9 ff a9 18  71
c130 : 20 d2 ff a9 0d 20 d2 ff  0b
c138 : 20 cc ff a2 04 20 c9 ff  7b
c140 : a2 19 20 e1 ff f0 24 20  0b
c148 : df c2 a0 00 b1 fe 20 0f  62
c150 : c3 20 d2 ff c8 c0 28 d0  ad
c158 : f3 a9 0d 20 d2 ff 98 18  27
c160 : 65 fe 85 fe 90 02 e6 ff  3a
c168 : ca d0 d7 20 cc ff a2 06  f8
c170 : 20 c9 ff a9 24 20 d2 ff  38
c178 : a9 0d 20 d2 ff 20 cc ff  3e
c180 : a2 06 20 c3 ff a9 05 20  47
c188 : c3 ff a9 04 20 c3 ff 4c  ee
c190 : e7 ff 78 a9 34 85 01 a9  91
c198 : 00 aa 9d 34 03 e8 e0 08  e6
c1a0 : d0 f8 a9 80 85 02 a0 00  52
c1a8 : b1 fe a2 00 0a 90 0a 48  5f
c1b0 : bd 34 03 05 02 9d 34 03  cd
c1b8 : 68 e8 e0 08 d0 ee 46 02  6f
c1c0 : c8 c0 08 d0 e3 a9 37 85  78
c1c8 : 01 58 a5 fc f0 0f a2 00  10
c1d0 : bd 34 03 49 ff 9d 34 03  55
c1d8 : e8 e0 08 d0 f3 a9 00 aa  2e
c1e0 : dd 34 03 d0 05 e8 e0 08  dd
c1e8 : d0 f6 60 a2 05 20 c9 ff  18
c1f0 : a0 00 b9 34 03 20 d2 ff  02
c1f8 : c8 c0 08 d0 f5 a9 0d 20  5e
```

Listing 1. »HC MPS-802«, die Super-Hardcopy geben Sie bitte mit dem MSE ein

```
c200 : d2 ff 4c cc ff a2 04 20  e4
c208 : c9 ff 18 a5 fd 65 71 aa  b2
c210 : f0 08 a9 20 20 d2 ff ca  a1
c218 : d0 fa a9 fe 20 d2 ff a9  9b
c220 : 8d 20 d2 ff 4c cc ff 18  cd
c228 : a5 fe 69 08 85 fe 90 02  3e
c230 : e6 ff e6 71 a5 71 c9 28  5b
c238 : 60 a2 04 20 c9 ff a9 0d  4c
c240 : 20 d2 ff 4c cc ff e6 72  a0
c248 : a5 72 c9 19 60 a2 04 20  27
c250 : c9 ff a9 0e 20 d2 ff a5  29
c258 : 71 aa f0 08 a9 20 20 d2  1d
c260 : ff ca d0 fa a9 fe 20 d2  10
c268 : ff a9 8d 20 d2 ff 4c cc  9b
c270 : ff 20 d2 c2 a2 00 a9 c0  df
c278 : 85 02 b9 34 03 18 0a 90  2d
c280 : 0a 48 a5 02 19 40 03 99  2b
c288 : 40 03 68 46 02 46 02 e8  59
c290 : e0 04 d0 e9 c8 c0 08 d0  38
c298 : db 4c c4 c2 20 d2 c2 a2  0c
c2a0 : 00 a9 03 85 02 b9 34 03  ab
c2a8 : 18 4a 90 0a 48 a5 02 19  37
c2b0 : 40 03 99 40 03 68 06 02  70
c2b8 : 06 02 e8 e0 04 d0 e9 c8  15
c2c0 : c0 08 d0 db a0 00 b9 40  a5
c2c8 : 03 99 34 03 c8 c0 08 d0  5a
c2d0 : f5 60 a9 00 aa 9d 40 03  fe
c2d8 : e8 e0 08 d0 f8 a8 60 a5  ee
c2e0 : fd f0 09 a8 a9 20 20 d2  6e
c2e8 : ff 88 d0 fa ad 18 d0 29  f0
c2f0 : 02 c9 02 d0 05 a9 11 20  94
c2f8 : d2 ff a5 fb c9 02 d0 05  ad
c300 : a9 0e 20 d2 ff a5 fc f0  16
c308 : 05 a9 12 20 d2 ff 60 29  6b
c310 : 7f 48 29 20 d0 0a 68 48  91
c318 : 29 40 d0 0c 68 09 40 60  a8
c320 : 68 48 29 40 d0 06 68 60  9e
c328 : 68 09 20 60 68 29 3f 09  08
c330 : 80 60 00 ff 00 ff 00 ff  e0
```

Listing 1. »HC MPS-802« (Schluß)

```
1000 -.ba$c000
1005 -          jsr ctrl
1010 -          cpx #$03
1015 -          bcs illegal
1020 -          stx $fb
1025 -          jsr ctrl
1030 -          cpx #$02
1035 -          bcs illegal
1040 -          stx $fc
1045 -          ldy #$00
1050 -          sty $fd
1055 -          ldy $fb
1060 -          cpy #$02
1065 -          beq start
1070 -          jsr ctrl
1075 -          cpx #$29
1080 -          bcs illegal
1085 -          stx $fd
1090 -          jmp start
1095 -ctrl      clc
1100 -          jsr $aefd
1105 -          jmp $b79e
1110 -illegal   jmp $b248
1115 -start     lda $dd02
1120 -          ora #$03
1125 -          sta $dd02
1130 -          jsr $ffe7
1135 -          lda #$04
1140 -          ldx #$04
1145 -          ldy #$00
1150 -          jsr open
1155 -          lda #$05
1160 -          ldx #$04
1165 -          ldy #$05
1170 -          jsr open
1175 -          lda #$06
1180 -          ldx #$04
1185 -          ldy #$06
1190 -          jsr open
1195 -          ldx #$06
1200 -          jsr $ffc9
1205 -          lda #$15
1210 -          jsr $ffd2
1215 -          lda #$0d
1220 -          jsr $ffd2
1225 -          jsr $ffcc
1230 -          lda #$00
1235 -          sta $fe
1240 -          sta $72
1245 -          lda $dd00
1250 -          eor #$ff
1255 -          and #$03
1260 -          clc
1265 -          ror
1270 -          ror
1275 -          ror
1280 -          sta $ff
1285 -          lda $d011
1290 -          and #$20
1295 -          cmp #$20
1300 -          bne lores
1305 -          lda $d018
1310 -          and #$08
1315 -          asl
1320 -          asl
1325 -          ora $ff
1330 -          sta $ff
1335 -          jmp hires
1340 -open      jsr $ffba
1345 -          lda #$00
1350 -          jsr $ffbd
1355 -          jmp $ffc0
1360 -lores     jmp lowres
1365 -hires     lda $fb
1370 -          cmp #$02
1375 -          beq hires4
1380 -hires1    lda #$00
1385 -          sta $71
1390 -new       jsr $ffe1
1395 -          beq end1
1400 -          jsr himode
1405 -          beq next
1410 -          jsr print5
1415 -          jsr print1
1420 -next      jsr char
1425 -          bne new
1430 -          jsr carrige
1435 -          jsr line
1440 -          bne hires1
1445 -end1      jmp close
1450 -hires4    lda $ff
1455 -          pha
1460 -          lda $fe
1465 -          pha
1470 -          lda #$00
1475 -          sta $71
1480 -line1     jsr $ffe1
1485 -          beq end4
1490 -          jsr himode
1495 -          beq next1
1500 -          jsr part1
1505 -          jsr print5
1510 -          jsr print4
1515 -next1     jsr char
1520 -          bne line1
1525 -          jsr carrige
1530 -          pla
1535 -          sta $fe
1540 -          pla
1545 -          sta $ff
1550 -          lda #$00
1555 -          sta $71
1560 -line2     jsr $ffe1
1565 -          beq end4
1570 -          jsr himode
1575 -          beq next2
1580 -          jsr part2
1585 -          jsr print5
1590 -          jsr print4
1595 -next2     jsr char
1600 -          bne line2
1605 -          jsr carrige
1610 -          jsr line
1615 -          bne hires4
1620 -end4      jmp close
1625 -lowres    lda $d018
1630 -          and #$f0
1635 -          lsr
1640 -          lsr
1645 -          ora $ff
1650 -          sta $ff
1655 -          ldx #$06
1660 -          jsr $ffc9
1665 -          lda #$18
1670 -          jsr $ffd2
1675 -          lda #$0d
1680 -          jsr $ffd2
1685 -          jsr $ffcc
1690 -          ldx #$04
1695 -          jsr $ffc9
1700 -          ldx #$19
1705 -zeile     jsr $ffe1
1710 -          beq close
1715 -          jsr lowmode
1720 -          ldy #$00
1725 -zeichen   lda ($fe),y
1730 -          jsr ascii
1735 -          jsr $ffd2
1740 -          iny
1745 -          cpy #$28
1750 -          bne zeichen
1755 -          lda #$0d
1760 -          jsr $ffd2
1765 -          tya
1770 -          clc
1775 -          adc $fe
1780 -          sta $fe
1785 -          bcc again
1790 -          inc $ff
1795 -again     dex
1800 -          bne zeile
1805 -close     jsr $ffcc
1810 -          ldx #$06
1815 -          jsr $ffc9
1820 -          lda #$24
1825 -          jsr $ffd2
1830 -          lda #$0d
1835 -          jsr $ffd2
1840 -          jsr $ffcc
1845 -          ldx #$06
```

Listing 2. Das Quell-Code-Listing zu »HC MPS-802«. Sollte sich die Hardcopy-Routine mit einem anderen Programm überschneiden, können Sie es damit in einen anderen Bereich assemblieren lassen.

```
1850 -         jsr $ffc3
1855 -         lda #$05
1860 -         jsr $ffc3
1865 -         lda #$04
1870 -         jsr $ffc3
1875 -         jmp $ffe7
1880 -himode   sei
1885 -         lda #$34
1890 -         sta $01
1895 -         lda #$00
1900 -         tax
1905 -clear    sta $0334,x
1910 -         inx
1915 -         cpx #$08
1920 -         bne clear
1925 -         lda #$80
1930 -         sta $02
1935 -         ldy #$00
1940 -move     lda ($fe),y
1945 -         ldx #$00
1950 -turn     asl
1955 -         bcc weiter
1960 -         pha
1965 -         lda $0334,x
1970 -         ora $02
1975 -         sta $0334,x
1980 -         pla
1985 -weiter   inx
1990 -         cpx #$08
1995 -         bne turn
2000 -         lsr $02
2005 -         iny
2010 -         cpy #$08
2015 -         bne move
2020 -         lda #$37
2025 -         sta $01
2030 -         cli
2035 -         lda $fc
2040 -         beq space
2045 -         ldx $00
2050 -revers   lda $0334,x
2055 -         eor #$ff
2060 -         sta $0334,x
2065 -         inx
2070 -         cpx #08
2075 -         bne revers
2080 -space    lda #$00
2085 -         tax
2090 -zero     cmp $0334,x
2095 -         bne return
2100 -         inx
2105 -         cpx #$08
2110 -         bne zero
2115 -return   rts
2120 -print5   ldx #$05
2125 -         jsr $ffc9
2130 -         ldy #$00
2135 -print    lda $0334,y
2140 -         jsr $ffd2
2145 -         iny
2150 -         cpy #$08
2155 -         bne print
2160 -         lda #$0d
2165 -         jsr $ffd2
2170 -         jmp $ffcc
2175 -print1   ldx #$04
2180 -         jsr $ffc9
2185 -         clc
2190 -         lda $fd
2195 -         adc $71
2200 -         tax
2205 -         beq goon
2210 -         lda #$20
2215 -goto     jsr $ffd2
2220 -         dex
2225 -         bne goto
2230 -goon     lda #$fe
2235 -         jsr $ffd2
2240 -         lda #$8d
2245 -         jsr $ffd2
2250 -         jmp $ffcc
2255 -char     clc
2260 -         lda $fe
2265 -         adc #$08
2270 -         sta $fe
2275 -         bcc help
2280 -         inc $ff
2285 -help     inc $71
2290 -         lda $71
2295 -         cmp #$28
2300 -         rts
2305 -carrige  ldx #$04
2310 -         jsr $ffc9
2315 -         lda #$0d
2320 -         jsr $ffd2
2325 -         jmp $ffcc
2330 -line     inc $72
2335 -         lda $72
2340 -         cmp #$19
2345 -         rts
2350 -print4   ldx #$04
2355 -         jsr $ffc9
2360 -         lda #$0e
2365 -         jsr $ffd2
2370 -         lda $71
2375 -         tax
2380 -         beq out
2385 -         lda #$20
2390 -rueck    jsr $ffd2
2395 -         dex
2400 -         bne rueck
2405 -out      lda #$fe
2410 -         jsr $ffd2
2415 -         lda #$8d
2420 -         jsr $ffd2
2425 -         jmp $ffcc
2430 -part1    jsr clm
2435 -save1    ldx #$00
2440 -         lda #$c0
2445 -         sta $02
2450 -         lda $0334,y
2455 -neu1     clc
2460 -         asl
2465 -         bcc old1
2470 -         pha
2475 -         lda $02
2480 -         ora $0340,y
2485 -         sta $0340,y
2490 -         pla
2495 -old1     lsr $02
2500 -         lsr $02
2505 -         inx
2510 -         cpx #$04
2515 -         bne neu1
2520 -         iny
2525 -         cpy #$08
2530 -         bne save1
2535 -         jmp change
2540 -part2    jsr clm
2545 -save2    ldx #$00
2550 -         lda #$03
2555 -         sta $02
2560 -         lda $0334,y
2565 -neu2     clc
2570 -         lsr
2575 -         bcc old2
2580 -         pha
2585 -         lda $02
2590 -         ora $0340,y
2595 -         sta $0340,y
2600 -         pla
2605 -old2     asl $02
2610 -         asl $02
2615 -         inx
2620 -         cpx #$04
2625 -         bne neu2
2630 -         iny
2635 -         cpy #$08
2640 -         bne save2
2645 -change   ldy #$00
2650 -chagain  lda $0340,y
2655 -         sta $0334,y
2660 -         iny
2665 -         cpy #$08
2670 -         bne chagain
2675 -         rts
2680 -clm      lda #$00
2685 -         tax
2690 -clr      sta $0340,x
2695 -         inx
2700 -         cpx #$08
2705 -         bne clr
2710 -         tay
2715 -         rts
2720 -lowmode  lda $fd
2725 -         beq shift
2730 -         tay
2735 -         lda #$20
2740 -tab      jsr $ffd2
2745 -         dey
2750 -         bne tab
2755 -shift    lda $d018
2760 -         and #$02
2765 -         cmp #$02
2770 -         bne gross
2775 -         lda #$11
2780 -         jsr $ffd2
2785 -gross    lda $fb
2790 -         cmp #$02
2795 -         bne invers
2800 -         lda #$0e
2805 -         jsr $ffd2
2810 -invers   lda $fc
2815 -         beq rts
2820 -         lda #$12
2825 -         jsr $ffd2
2830 -rts      rts
2835 -ascii    and #$7f
2840 -         pha
2845 -         and #$20
2850 -         bne cont
2855 -         pla
2860 -         pha
2865 -         and #$40
2870 -         bne up1
2875 -         pla
2880 -         ora #$40
2885 -         rts
2890 -cont     pla
2895 -         pha
2900 -         and #$40
2905 -         bne up2
2910 -         pla
2915 -         rts
2920 -up1      pla
2925 -         ora #$20
2930 -         rts
2935 -up2      pla
2940 -         and #$3f
2945 -         ora #$80
2950 -         rts
```

Listing 2. Quell-Code-Listing zu »HC MPS-802« (Schluß)

64er ONLINE

Grund genug fürs neue 64'er: Die brandaktuellen Informationen zum Thema Datenfernübertragung!

★ **Was bringt uns BTX?**
Softwareeinkauf per Bildschirm.
Weltweiter Datenverkehr zum Billigtarif mit Datex P.

★ **Im Test:**
Akustikkoppler und Terminalprogramme.

★ **Großer Testbericht:**
Was leistet der Amiga? Lohnt sich der Kauf?

★ **Künstliche Intelligenz:**
»Prolog 64« im Test.

★ **Die wichtigsten Programmiersprachen für den C64.**
Ihre Vorteile – ihre Nachteile.
Pascal-Kurs für Anfänger.
Ist »C« nur eine Sprache für Profis?
Was ist ein Compiler?

64ER ONLINE

★ **Schon getestet:**
Die ersten Mäuse für den C64 von Rushware und NCE.

★ **...und natürlich – wie in jeder Ausgabe – jede Menge Tips & Tricks für den C64 und C128.**

Das März-Heft erhalten Sie ab 14.2.86 überall im Zeitschriftenhandel.

Falls Sie 64'er noch nicht regelmäßig beziehen, sichern Sie sich jetzt Ihr persönliches Abonnement.

Wenn Sie zur Anforderung den nebenstehenden Gutschein benutzen, genießen sie nicht nur die Abonnentenvorzüge, sondern erhalten außerdem die neueste Ausgabe kostenlos und unverbindlich als Probeheft!



FÜR EIN KOSTENLOSES PROBEEXEMPLAR DES 64'er-MAGAZINS

JA, ich möchte das »64'er-Magazin« kennenlernen.
Senden Sie mir bitte die aktuellste Ausgabe kostenlos als Probeexemplar. Wenn mir »64'er« gefällt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte »64'er« dann regelmäßig frei Haus per Post und bezahle pro Jahr DM 78,— (Ausland auf Anfrage)

Vorname, Name

Straße PLZ, Ort

Datum 1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum 2. Unterschrift

Gutschein ausfüllen, ausschneiden, in ein Kuvert stecken und absenden an: Markt&Technik Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar

64SO286

64'er-online.de
64er-online.net

Disketten-Reparatur mit Reformat

Haben Sie auch schon einmal versehentlich eine Diskette neu formatiert? Wenn ja, dann brauchen Sie »Reformat«. Denn mit diesem Programm läßt sich die Diskette sehr einfach »reparieren«.

Wenn eine bespielte Diskette ohne Angabe einer ID formatiert wurde, präsentiert sich das Directory in jungfräulichem Zustand mit friedlichen 646 freien Blöcken. Doch dieser Zustand trügt, denn in Wirklichkeit sind noch alle Daten auf der Diskette vorhanden, nur der Zugang dazu ist verbaut. Es werden bei der Formatierung ohne ID nur die BAM (18,0) und der erste Sektor der Directory (18,1) formatiert. Deshalb dauert dieser Vorgang auch nur einige Sekunden, während die Formatierung mit ID, bei der alle Blöcke formatiert werden, eine halbe Ewigkeit braucht. Da in jedem Sektor der Directory 8 Files verwaltet werden, sind außer den ersten 8 Filenamen alle weiteren Filenamen noch vorhanden.

Um die Programme auf der Diskette zu retten, muß man »nur« die gelöschten Sektoren 18,0 und 18,1 in der Directory wieder rekonstruieren. Man benötigt dazu vor allem die Start-Blöcke der Programme.

Um die Start-Blöcke zu finden, müssen die ersten beiden Byte aller Blöcke von der Diskette eingelesen werden. Da diese Byte die Zeiger auf die folgenden Blöcke enthalten, kann man durch Zuordnen die Programmanfänge herausfinden. Diesen Vorgang erledigt »Reformat«.

Die Bedienung

Zum Ausprobieren legen wir uns eine »absichtlich« formatierte Diskette zu. Machen Sie sich eine Kopie von einer Diskette mit mehr als 8 Programmen. Diese Kopie formatieren Sie mit OPEN 1,8,15,"N:TEST".

Wenn Sie nun das Directory dieser Diskette betrachten, haben Sie ein leeres Inhaltsverzeichnis auf dem Bildschirm.

Nun laden Sie »Reformat« (siehe Listing) und legen nach dem Start mit RUN die Test-Diskette in das Laufwerk.

Sie befinden sich nun im Auswahlmenü. Mit F7 können Sie die Sektoren 1 bis 18 der Spur 18 untersuchen. Es werden in diesem Programmpunkt die untersuchten Blöcke mit den Zeigern auf die nächsten Blöcke aufgelistet. Zeigt ein Sektor auf 0,255, so ist dieser Sektor der letzte Sektor mit Einträgen. Die Reihenfolge der Sektoren ist dabei nicht linear (1,2,3,4,..), sondern geht in Dreierschritten von 1 bis 18 (1,4,7,10,13,16;2,5,8,11,14,17;3,6,9...). Zeigt ein Sektor auf 75,1, dann ist dieser Sektor nicht benutzt, also noch im frisch formatierten Zustand und hat deshalb keine Daten gespeichert.

»NO MORE FILES IN SEKTOR x« weist darauf hin, daß in diesem Sektor nicht alle 8 Einträge vorhanden sind.

Hatte das Directory nicht mehr als 8 Einträge, so wird man keine Programmnamen mehr finden; hatte sie jedoch mehr als 8 Einträge, so sind ab Sektor 4 (dem Folge-Sektor vom gelöschten Sektor 1) die restlichen Einträge mit Namen, Anzahl der Blöcke, Programmtyp und Startspur und -sektor angegeben.

Diesen Programmpunkt kann man durch Drücken der »←«-Taste vorzeitig verlassen. Am Schluß dieses Programmteils kommt man durch einen Tastendruck wieder ins Auswahlmenü.

Mit F1 wird die eigentliche Rekonstruktion des Directory gestartet. Es werden zuerst die Block-Zeiger eingelesen.

Als nächstes werden die kritischen Blöcke, das heißt die Blöcke, auf die mehrere Blöcke zeigen, aufgelistet. Diese Blöcke sollte man sich notieren. Nun muß man entscheiden, ob man die noch vorhandenen Programmnamen im Directory für die Rekonstruktion hernehmen will. Wenn ja, dann wird zusätzlich das Directory auf noch vorhandene Namen hin untersucht. Im Anschluß daran werden die gefundenen Programme aufgelistet. Man hat nun zu entscheiden, welche Programme man wieder übernehmen möchte. Beantwortet man die Frage »WRITE BACK IN DIRECTORY« mit ja, dann wird das Programm in die BAM eingetragen.

Hierbei können einige Probleme auftreten. Kritische Blöcke sollten nicht in das Directory zurückgeschrieben werden, da Blöcke, die als belegt gekennzeichnet werden sollten und schon belegt sind, die Fehlermeldung »no block« auslösen. In diesem Fall kann man durch das im Programm vorgeschlagene Drücken der F7-Taste sich durch die kritischen Blöcke hangeln, bis ein normales Programm gefunden wird. Wichtig ist, daß auf der zu reformatierenden Diskette 664 Blöcke frei sind, da sonst die gefundenen Programme nicht richtig in die BAM eingetragen werden können, was ebenfalls die Fehlermeldung »no block« zur Folge hat.

Wenn im Directory noch der alte Name zu finden war, wird dieser vom Programm vorgeschlagen, wenn der alte Name schon gelöscht war, muß ein neuer Name eingegeben werden.

Ist das gesuchte Programm eingetragen, läßt sich die Reformatierung ohne weiteres beenden.

Programme mit nur einem Block werden ignoriert.

Bei auftretenden Fehlern (zum Beispiel: read error oder no block) kann man das Programm durch F7 fortsetzen oder durch die ←-Taste neu starten. Bei der Fortsetzung mit F7 muß man sich unter Umständen durch mehrmaliges Drücken von F7 durch eine ganze Spur wurschteln.

Aus Geschwindigkeitsgründen ist es empfehlenswert, das Programm zu compilieren.

(G. Burger/ah)

```
1 REM ******************************        <139>
2 REM *         REFORMAT V4.0       *        <225>
3 REM *                             *        <052>
4 REM *   1985 BY GEORG BURGER      *        <138>
5 REM *     ROIDERSTRASSE 18        *        <144>
6 REM *       8051 ZOLLING          *        <209>
7 REM ******************************        <145>
8 :                                          <240>
9 :                                          <241>
10 POKE 53280,0:POKE 53281,0:POKE 646,5:PR
   INT CHR$(142)                             <119>
20 OPEN 1,8,15,"I":CLOSE 1:GOTO 1000         <170>
97 :                                         <073>
98 REM  ----------ROUTINEN---------------   <217>
99 :                                         <075>
100 CLOSE 5:OPEN 5,8,5,"#2":RETURN          <199>
200 INPUT#15,Y1$,Y2$,Y3$,Y4$                <021>
210 IF VAL(Y1$)=0 THEN RETURN               <172>
220 PRINT"{DOWN}"Y1$" "Y2$" "Y3$" "Y4$      <092>
230 PRINT"{DOWN,3SPACE}USE{SPACE,RVSON,SPA
    CE}←{SPACE,RVOFF,SPACE}TO EXIT OR{SPAC
    E,RVSON,SPACE}F7{SPACE,RVOFF,SPACE}TO
    CONTINUE{DOWN}"                          <161>
240 POKE 198,0                               <148>
250 GET A$:IF A$=""THEN 250                  <220>
260 IF A$="←"THEN RUN                        <168>
270 IF ASC(A$)=136 THEN RETURN               <151>
280 GOTO 250                                 <042>
```

Listing zum Programm »Reformat«. Bitte mit dem Checksummer eingeben.

```
300 PRINT"{CLR}@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@@@@@@@@@@";                           <225>
310 PRINT"{RVSON,3SPACE}REFORMAT{2SPACE}V4
    .0{4SPACE}BY{2SPACE}GEORG BURGER{3SPAC
    E,RVOFF}";                                <183>
320 FOR I=1 TO 7:PRINT"G{38SPACE}H";:NEXT
    I                                         <134>
330 PRINT"TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
    TTTTTTTT":RETURN                          <187>
400 PRINT"OLD DIRECTORY-REGISTER:":PRINT      <208>
410 PRINT"NAME",DN$(D1)                       <042>
420 PRINT"BLOCKS",DL(D1)                      <148>
430 PRINT"TRACK",DS(D1)                       <020>
440 PRINT"SECTOR",DB(D1)                      <031>
450 EF$="Y":RETURN                            <167>
500 PRINT"{DOWN,13SPACE,RVSON,SPACE}PRESS
    ANY KEY{SPACE,RVOFF}":POKE 198,0          <046>
510 GET A$:IF A$=""THEN 510                   <065>
520 RETURN                                    <070>
600 D1=0                                      <039>
610 IF(DS(D1)=AS)AND(DB(D1)=AB)THEN 400       <073>
620 D1=D1+1:IF D1>=DP THEN RETURN             <067>
630 GOTO 610                                  <106>
700 MB=NB:NB=NB+3                             <251>
710 IF NB=19 THEN NB=2                        <208>
720 IF NB=20 THEN NB=3                        <166>
730 RETURN                                    <024>
997 :                                         <211>
998 REM --------- MENUE ---------------       <088>
999 :                                         <213>
1000 OPEN 15,8,15:OPEN 5,8,5,"#2"             <001>
1010 DIM S(35,21),B(35,21),BL(35),DI$(29),
     DI(29),A$(255)                           <139>
1020 DIM SV(35,21),BV(35,21),US(35,21),DS(
     144),DB(144),DL(144),DN$(144),DT(144)    <190>
1030 FOR I=1 TO 17:BL(I)=20:NEXT:FOR I=18
     TO 24:BL(I)=18:NEXT                      <192>
1040 FOR I=25 TO 30:BL(I)=17:NEXT:FOR I=31
      TO 35:BL(I)=16:NEXT                     <248>
1050 GOSUB 300:POKE 214,4:POKE 211,0:SYS 5
     8640                                     <161>
1060 PRINT"{RIGHT,3SPACE,RVSON,SPACE}F1{SP
     ACE,RVOFF,3SPACE}RECONSTRUCT DIRECTOR
     Y                                        <156>
1070 PRINT"{RIGHT,3SPACE,RVSON,SPACE}F7{SP
     ACE,RVOFF,3SPACE}EXAMINE DIRECTORY":P
     OKE 198,0                                <119>
1080 GET A$:IF A$=""THEN 1080                 <245>
1090 IF ASC(A$)=133 THEN GOSUB 100:GOTO 20
     00                                       <109>
1100 IF ASC(A$)=136 THEN GOSUB 100:GOSUB 4
     000:GOTO 1050                            <035>
1110 GOTO 1080                                <035>
1997 :                                        <195>
1998 REM --- RECONSTRUCT DIRECTORY ----       <012>
1999 :                                        <197>
2000 GOSUB 300:POKE 214,4:POKE 211,0:SYS 5
     8640                                     <093>
2010 PRINT"{RIGHT,9SPACE}RECONSTRUCT DIREC
     TORY                                     <107>
2020 PRINT"{RIGHT,12SPACE}USE{SPACE,RVSON,
     SPACE}←{SPACE,RVOFF,SPACE}TO EXIT        <192>
2030 PRINT"{2DOWN}@@@@@@@@@@@@@@@@@@@@@@@@
     @@@@@@@@@@@@@@@@";                       <188>
2040 PRINT"{RVSON,SPACE}TRACK{3SPACE}SECTO
     R{3SPACE}TO{3SPACE}TRACK{5SPACE}SECTO
     R{SPACE,RVOFF}"                          <199>
2050 FOR S=1 TO 35:FOR B=0 TO BL(S)           <051>
2060 PRINT#15,"B-R:"5;0;S;B:GOSUB 200         <154>
2070 PRINT#15,"B-P:"5;0:GOSUB 200             <109>
2080 GET#5,E$                                 <243>
2090 GET#5,F$                                 <084>
2100 E=ASC(E$+CHR$(0))                        <211>
2110 F=ASC(F$+CHR$(0))                        <033>
2120 PRINT"{2SPACE}"S,B,"{4SPACE}"E,"{4SPA
     CE}"F                                    <090>
2130 GET A$:IF A$=""THEN 2150                 <245>
2140 IF A$="←"THEN CLOSE 15:RUN               <043>
2150 IF E=75 AND F=1 THEN US(S,B)=-2:GOTO
     2200                                     <051>
2160 IF E>35 THEN E=0                         <227>
2170 IF F>21 THEN F=21                        <102>
2180 S(S,B)=E:B(S,B)=F                        <245>
2190 SV(E,F)=S:BV(E,F)=B:US(E,F)=US(E,F)+1    <214>
2200 NEXT:NEXT                                <171>
2210 REM ------ CRITICAL BLOCKS -------       <114>
2220 GOSUB 300:POKE 214,5:POKE 211,0:SYS 5
     8640                                     <075>
2230 PRINT"{RIGHT,12SPACE}CRITICAL BLOCKS     <151>
2240 PRINT"{4DOWN,RVSON,SPACE}TRACK{3SPACE
     }SECTOR{3SPACE}LENGHT{SPACE,RVOFF}":P
     RINT                                     <227>
2250 FOR S=1 TO 35                            <194>
2260 FOR B=0 TO BL(S)                         <169>
2270 IF US(S,B)<2 THEN 2290                   <245>
2280 PRINT"{2SPACE}"S,B,US(S,B)               <105>
2290 NEXT:NEXT                                <005>
2300 PRINT"{DOWN,9SPACE,RVSON,SPACE}END OF
      CRITICAL BLOCKS{SPACE,RVOFF}"           <148>
2310 GOSUB 500:GOSUB 300:POKE 214,4:POKE 2
     11,0:SYS 58640                           <181>
2320 PRINT"{RIGHT,3SPACE}USE OLD DIRECTORY
      TO NAME FILES ?                         <243>
2330 PRINT"{RIGHT,16SPACE}(Y/N)               <006>
2340 GET DA$:IF DA$=""THEN 2340               <119>
2350 IF DA$<>"N"THEN DA$="Y":GOSUB 4030       <157>
2360 FOR S=1 TO 35                            <050>
2370 IF S=18 THEN 2550                        <063>
2380 FOR B=0 TO BL(S)                         <035>
2390 S1=S:B1=B                                <033>
2400 IF(S=1)AND(S(S,B)=0)AND(B(S,B)=1)THEN
      2540                                    <231>
2410 IF US(S1,B1)<0 THEN 2540                 <087>
2420 IF US(S1,B1)>0 THEN S2=SV(S1,B1):B2=B
     V(S1,B1):S1=S2:B1=B2:GOTO 2420           <173>
2430 PRINT"{CLR}***************{RVSON,SPACE
     }FILE-START{SPACE,RVOFF}**************
     *";:BA=1:AS=S1:AB=B1                     <184>
2440 PRINT S1,B1:US(S1,B1)=-1                 <016>
2450 S2=S(S1,B1):B2=B(S1,B1):S1=S2:B1=B2      <235>
2460 IF S1<>0 THEN BA=BA+1:GOTO 2440          <213>
2470 PRINT"****************{RVSON,SPACE}FIL
     E-END{SPACE,RVOFF}****************{DOW
     N}"                                      <119>
2475 PRINT"LENGHT:";BA;"BLOCKS":PRINT         <231>
2480 IF BA<2 THEN 2540                        <216>
2490 IF DA$="Y"THEN EF$="N":GOSUB 600         <214>
2500 PRINT"{DOWN}WRITE BACK IN DIRECTORY (
     Y/N)":POKE 198,0:PRINT                   <127>
2510 GET E$:IF E$=""THEN 2510                 <148>
2520 IF E$="Y"THEN 2570                       <105>
2530 IF E$<>"N"THEN 2510                      <199>
2540 NEXT B                                   <026>
2550 NEXT S                                   <172>
2560 GOTO 1050                                <120>
2570 REM -------- WRITE BAM -----------       <236>
2580 S1=AS:B1=AB                              <222>
2590 PRINT#15,"B-A:"0;S1;B1:GOSUB 200         <178>
2600 S2=S(S1,B1):B2=B(S1,B1):S1=S2:B1=B2      <131>
2610 IF S1<>0 THEN 2590                       <198>
2620 REM ----- WRITE DIRECTORY --------       <074>
2630 MB=1:NB=4                                <030>
2640 PRINT#15,"U1:"5;0;18;MB:GOSUB 200        <107>
2650 PRINT#15,"B-P:"5;0:GOSUB 200             <181>
2660 GET#5,NS$,NB$                            <052>
2670 NS=ASC(NS$+CHR$(0))                      <114>
2680 IF NS<>0 THEN GOSUB 700:GOTO 2640        <157>
2690 C=0                                      <059>
2700 GET#5,A$,B$,C$                           <210>
2710 IF A$=""THEN A$="0"                      <017>
2720 IF B$=""THEN B$="0"                      <037>
2730 IF C$=""THEN C$="0"                      <057>
2740 IF A$="0"THEN IF B$="0"THEN IF C$="0"
     THEN 2910                                <045>
2750 FOR I=1 TO 27:GET#5,A$:NEXT              <120>
2760 C=C+1:IF C<8 THEN GET#5,A$,A$:GOTO 27
     00                                       <138>
2770 IF MB<18 THEN 2810                       <247>
2780 PRINT"DIRECTORY IS FULL !"               <194>
2790 GET A$:IF A$=""THEN 2790                 <016>
2800 GOTO 1050                                <104>
2810 PRINT#15,"B-P:"5;0:GOSUB 200             <085>
2820 PRINT#5,CHR$(18);                        <247>
2830 PRINT#5,CHR$(NB);                        <154>
2840 PRINT#15,"U2:"5;0;18;MB:GOSUB 200        <054>
2850 GOSUB 700                                <048>
2860 PRINT#15,"U1:"5;0;18;MB:GOSUB 200        <073>
2870 PRINT#15,"B-P:"5;0:GOSUB 200             <147>
2880 PRINT#5,CHR$(0);CHR$(255);:FOR I=0 TO
      253:PRINT#5,CHR$(0);:NEXT               <227>
2890 PRINT#15,"U2:"5;0;18;MB:GOSUB 200        <104>
2900 GOTO 2640                                <022>
2910 PRINT"TRACK 18 SECTOR"MB"POSITION"C+1
     :PRINT                                   <119>
2920 P=2+C*32                                 <137>
```

Listing zum Programm »Reformat«

```
2930 PRINT#15,"B-P:"5;P:GOSUB 200                <215>
2940 DI$(0)=CHR$(130)                            <193>
2950 DI$(1)=CHR$(AS)                             <099>
2960 DI$(2)=CHR$(AB)                             <156>
2970 FOR I=3 TO 18:DI$(I)=CHR$(160):NEXT        <066>
2980 IF DA$="Y"AND EF$="Y"THEN PRINT"{7SPA
     CE}"DN$(D1)"{UP}"                           <135>
2990 INPUT"NAME ";N$                             <222>
3000 N$=LEFT$(N$,16)                             <032>
3010 FOR I=0 TO LEN(N$)-1                        <224>
3020 DI$(3+I)=MID$(N$,I+1,1)                     <197>
3030 NEXT                                        <246>
3040 FOR I=19 TO 27:DI$(I)=CHR$(0):NEXT          <140>
3050 BH=INT(BA/256):BL=BA-256*BH                 <024>
3060 DI$(28)=CHR$(BL)                            <057>
3070 DI$(29)=CHR$(BH)                            <067>
3080 FOR I=0 TO 29                               <028>
3090 PRINT#5,DI$(I);                             <026>
3100 NEXT                                        <062>
3110 PRINT#15,"U2:"5;0;18;MB:GOSUB 200           <070>
3120 GOTO 2540                                   <228>
3997 :                                           <163>
3998 REM ---- EXAMINE DIRECTORY -------          <049>
3999 :                                           <165>
4000 GOSUB 300:POKE 214,4:POKE 211,0:SYS 5
     8640                                        <061>
4010 PRINT"{RIGHT,11SPACE}EXAMINE DIRECTOR
     Y                                           <162>
4020 PRINT"{RIGHT,12SPACE}USE{SPACE,RVSON,
     SPACE}←{SPACE,RVOFF,SPACE}TO EXIT           <160>
4030 PRINT                                       <066>
4040 MB=1:DP=0                                   <163>
4050 PRINT#15,"U1:"5;0;18;MB:GOSUB 200           <247>
4060 PRINT#15,"B-P:"5;0:GOSUB 200                <065>
4070 GET#5,NS$,NB$                               <192>
4080 NB=ASC(NB$+CHR$(0))                         <101>
4090 NS=ASC(NS$+CHR$(0))                         <008>
4100 PRINT"{DOWN}TTTTTTTTTTTTTTTTTTTTTTTTT
     TTTTTTTTTTTTTTT";                           <170>
4110 PRINT"{5SPACE}TRACK 18{2SPACE}SECTOR"
     MB"-->"NS" "NB                              <105>
4120 PRINT"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @@@@@@@@@";                                 <041>
4130 GET A$:IF A$=""THEN 4150                    <021>
4140 IF A$="←"THEN CLOSE 15:RUN                  <011>
4150 IF NS=75 AND NB=1 THEN 4440                 <147>
4160 PRINT                                       <198>
4170 PRINT"{RVSON}BLOCKS"TAB(7)"NAME"TAB(2
     6)"TYP"TAB(31)"TRA."TAB(36)"SEC.{RVOF
     F}"                                         <141>
4180 FOR C=0 TO 7                                <137>
4190 GET A$:IF A$=""THEN 4210                    <143>
4200 IF A$="←"THEN CLOSE 15:RUN                  <071>
4210 FOR I=0 TO 29                               <142>
4220 GET#5,DI$(I)                                <170>
4230 NEXT I                                      <250>
4240 IF C<>7 THEN GET#5,A$,A$                    <140>
4250 FOR I=0 TO 29:DI(I)=ASC(DI$(I)+CHR$(0
     )):NEXT                                     <081>
4260 KW$=" NO MORE FILES IN SECTOR"              <018>
4270 IF DI(0)=0 AND DI(1)=0 AND DI(2)=0 TH
     EN PRINT"{DOWN,6SPACE,RVSON}"KW$;MB"{
     LEFT,SPACE,RVOFF}":PRINT:GOTO 4440          <175>
4280 DS(DP)=DI(1):DB(DP)=DI(2)                   <175>
4290 DL(DP)=DI(29)*256+DI(28)                    <116>
4300 DN$(DP)="":FOR I=3 TO 18:IF DI(I)<>16
     0 THEN DN$(DP)=DN$(DP)+DI$(I)               <045>
4310 NEXT I                                      <074>
4320 DT(DP)=DI(0)                                <045>
4330 DP=DP+1                                     <234>
4340 PRINT DI(29)*256+DI(28);TAB(7);CHR$(3
     4);                                         <051>
4350 FOR I=3 TO 18:IF DI(I)<>160 THEN PRIN
     T DI$(I);                                   <185>
4360 NEXT:PRINT CHR$(34);                        <171>
4370 BY=DI(0)AND(NOT 128):IF BY=0 THEN PRI
     NT TAB(26);"DEL";                           <218>
4380 IF BY=1 THEN PRINT TAB(26);"SEQ";           <083>
4390 IF BY=2 THEN PRINT TAB(26);"PRG";           <021>
4400 IF BY=3 THEN PRINT TAB(26);"USR";           <248>
4410 IF BY=4 THEN PRINT TAB(26);"REL";           <168>
4420 PRINT TAB(30);DI(1);TAB(35);DI(2)           <208>
4430 NEXT C                                      <148>
4440 MB=MB+1:IF MB<>19 THEN 4050                 <135>
4450 IF DA$="Y"THEN RETURN                       <216>
4460 GOSUB 500:RETURN                            <075>
```

Listing zum Programm »Reformat« (Schluß)

Spline – das computergesteuerte Kurvenlineal

Durch wenige Stützstellen lassen sich beliebige Kurven sehr exakt ermitteln und darstellen. Daher ist das Programm für all diejenigen interessant, die in irgendeiner Form etwas mit der grafischen Auswertung von Meßwerten zu tun haben.

Die Motivation zum Erstellen einer ersten Version des Programms »SPLINE« erhielt ich während meiner Diplomarbeit. Große Mengen an Meßwerten waren innerhalb kurzer Zeit grafisch zu verarbeiten, um weiterführende Versuche planen zu können.

Die konventionelle Methode einer grafischen Versuchsauswertung besteht darin, die Meßwerte (Stützstellen) von Hand in ein entsprechend dimensioniertes Koordinatensystem einzutragen und in einem weiteren Arbeitsschritt mittels eines Kurvenlineals (Spline) zu einem stetigen Kurvenzug zu verbinden.

Durch den Einsatz des Programms »SPLINE« (Listing) in Verbindung mit Simons Basic, einem C 64, einer Floppy 1541 und einem Drucker MPS 801 verringert sich der Arbeitsaufwand auf das Eintippen der Wertepaare. Sind die Stützstellen erst einmal in einer Datei abgelegt, bestehen die verschiedensten Möglichkeiten, die Ergebniskurven zu kombinieren und darzustellen.

Ein Nachteil mit Vorteilen

Gemeint ist die bei zahlreichen Stützstellen erhebliche Rechenzeit, bedingt durch die reine Basic-Programmierung. Für alle jene, die einen Blick hinter die Kulissen werfen möchten, also den Berechnungsalgorithmus der Splines ergründen wollen, ist die Bearbeitung in Basic sicherlich leichter nachvollziehbar.

Was ist ein Spline?

Nur die im vorliegenden Programm verwendeten »kubischen Splines« sollen hier etwas näher erläutert werden. Der Graph einer kubischen Splinefunktion verhält sich wie eine Kurve, die mit einem elastischen Kurvenlineal (Spline) gezeichnet wird, so daß sie die vorgegebenen Punkte (Stützstellen) ohne Knick durchläuft.

Eine Splinefunktion besteht aus einer Anzahl von Teilfunktionen, die jeweils durch ein Polynom 3. Grades beschrieben werden.

$y = a + b * x + c * x^2 + d * x^3$

Die Anzahl dieser Polynome richtet sich nach der Anzahl der Stützstellen; je zwei Stützstellen werden durch ein Polynom verbunden (Bild 1).

Der glatte Übergang der Polynomgraphen an den Stützstellen hängt allein von der Bestimmung geeigneter Koeffizienten (a, b, c, d) ab. Zu diesem Zweck stellt man an den Berührpunkt zweier benachbarter Polynome vier Bedingungen:

64er-online.de

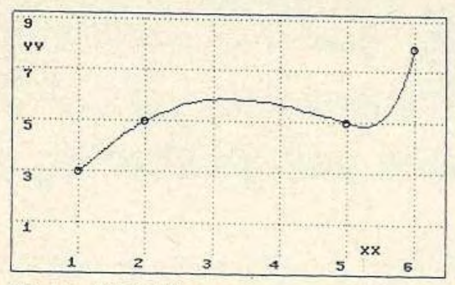


Bild 1. Mit Hilfe von nur 4 Stützstellen wird eine solche Kurve ermittelt

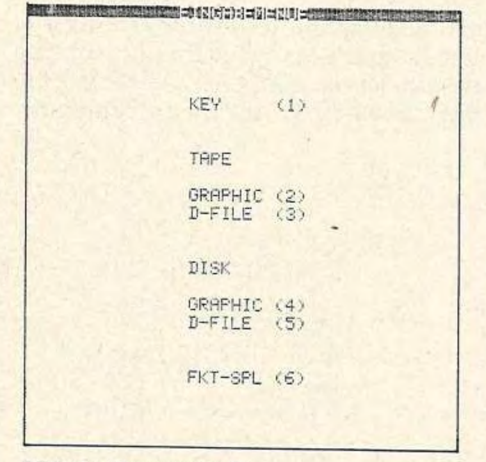


Bild 2. Hauptmenü

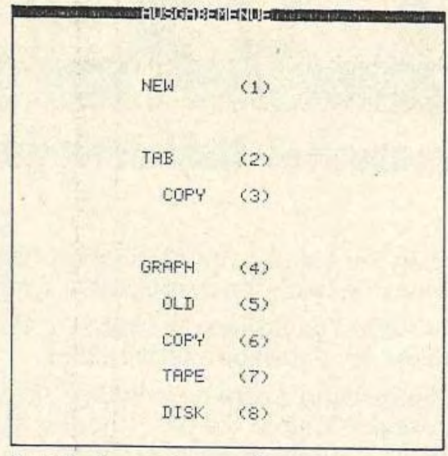


Bild 3. Ausgabemenü

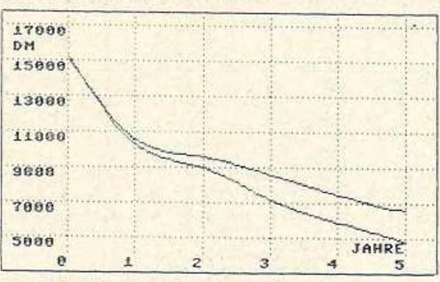


Bild 4. Wertverlust zweier Mittelklasse-Pkws

Bild 7. Beide Parabeln: deutlich erkennt man die geringe Abweichung

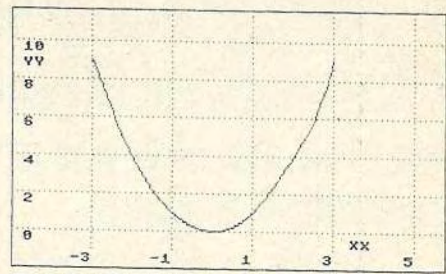


Bild 5. Mit »Spline« erzeugte Parabel

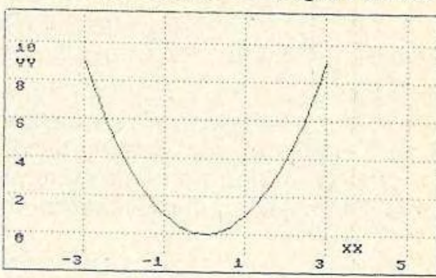
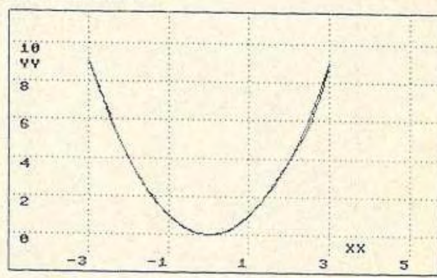


Bild 6. Tatsächlicher Kurvenverlauf der Parabel



1. Der Funktionswert des Polynoms entspricht an den Stützstellen dem y-Wert dieser Punkte.
2. Die y-Werte zweier Polynome im Berührpunkt sind gleich klar, sonst würden sie sich nicht berühren.
3. Die Steigung und...
4. die Krümmung der Polynomgraphen im Berührpunkt sollen gleich sein.

Aus diesen vier Bedingungen lassen sich für jedes Polynom vier Gleichungen ableiten – man erhält also pro Polynom ein Gleichungssystem. Ein Gleichungssystem, in dem die Anzahl der Gleichungen gleich der Zahl der Unbekannten ist, kann eindeutig gelöst werden. Die Unbekannten sind in unserem Fall die vier Koeffizienten a, b, c, d der Polynome.

Programmbeschreibung

Das Programm »SPLINE« arbeitet menügesteuert – die Eingabe erfolgt im Dialogbetrieb. Nach dem Start des Programms erscheint das Eingabemenü (Bild 2).

KEY
Eingabe eines neuen Datenfiles über die Tastatur.

TAPE-GRAPHIC
Abrufen einer vollständigen Grafik vom Band.

TAPE-D-FILE
Abrufen eines einzelnen (oder mehrerer) Datenfiles vom Band
DISK-GRAPHIC
DISK-D-FILE
siehe TAPE.

FKT-SPL.
Diese Option ermöglicht den Vergleich zwischen dem Graphen einer mathematischen Funktion und dem Kurvenverlauf eines durch Splines dargestellten Zusammenhangs.

KEY (1)
Nach Wahl der Option 1 erwartet das Programm Angaben über die einzugebenden Stützstellen beziehungsweise die nach der entsprechenden Splinefunktion zu berechnenden Zwischenwerte.

ANZAHL DER KURVEN,
deren Stützstellen eingegeben werden sollen.

ANZAHL DER INT.POL.SCHRITTE?
Anzahl der Interpolationsschritte, also Anzahl der Wertepaare, die vom Computer nach der ermittelten Splinefunktion zwischen zwei Stützstellen berechnet werden sollen. Bei einer zu geringen Zahl von Interpolationsschritten wird der Kurvenverlauf in der Grafik eckig – bei einer großen Anzahl erhöht sich die Rechenzeit.

EINGABE STÜTZSTELLEN
Die Eingabe der Stützstellen erfolgt entsprechend dem folgenden Beispiel:

Stützstellen		Anzeige	Eingabe
x	y		
1	2	X(1),Y(1)	1,2
1.5	2.4	X(2),Y(2)	1.5,2.4
3.75	5.12	X(3),Y(3)	3.75,5.12

FKT-SPL (6)
Nach Wahl der Option 6 erwartet das Programm die Eingabe einer mathematischen Funktion in Basic-Syntax.
Beispiele:

math. Fkt.	Eingabe
$f(x)=x^2$	X↑2
$f(x)=\sqrt{82x+3}$	SQR(82*X+3)

Des weiteren müssen als Funktionsdaten der Anfangs-x-Wert, der End-x-Wert und die Anzahl der Zwischenwerte innerhalb der Definitionsmenge (Anf.-x-Wert bis End-x-Wert) spezifiziert werden.

Ergänzen Sie jetzt Ihre 64'er-Sammlung

Schaffen Sie sich ein interessantes Nachschlagewerk und gleichzeitig ein wertvolles Archiv!

Kennen Sie alle Ausgaben von 64'er? Suchen Sie einen ganz bestimmten Testbericht? Oder haben Sie einen Teil eines interessanten Kurses versäumt? Suchen Sie nach einer speziellen Anwendung?

Damit Sie jetzt fehlende Hefte mit »Ihrem« Artikel nachbestellen können, finden Sie auf diesen Seiten eine Zusammenstellung aller wesentlichen Artikel der Ausgaben 01 bis 12/85.

Und so kommen Sie schnell an die noch lieferbaren Ausgaben: Prüfen Sie, welche Ausgabe in Ihrer Sammlung noch fehlt, oder welches Thema Sie interessiert. Tragen Sie die Nummer dieser Ausgabe und das Erscheinungsjahr (z.B. 2/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein. Die ausgefüllte Zahlkarte einfach heraustrennen und Rechnungsbetrag beim nächsten Postamt einzahlen. Ihre Bestellung wird nach Zahlungseingang umgehend zur Auslieferung gebracht.

Aktuell

Stichwort	Titel	Seite	Ausgabe
Inhalt	Jahresinhaltsverzeichnis 4/84 bis 3/85	84	04/85
Allgemeines	Commodore Gestern Heute Morgen	10	01/85
Computer	Amiga — Der neue Supercomputer	8	09/85
DFÜ	MCI Mail — Die schnelle Post	8	02/85
Interview	Interview mit David Crane (Game Designer)	146	06/85
Lernen	Schule braucht Computer (VAM-Computer)	9	06/85
Messen	International Chaos Communication Congress	15	03/85
	Heiße Messe in der Wüste: CES	8	03/85
	Die Sportler kommen (CES Software-Bericht)	9	04/85
	Musikmesse Frankfurt	8	04/85
	Hannover-Messe '85	8	06/85
	Hannover-Messe '85	8	07/85
	Chicago im Zeichen der CES	8	08/85
	Aktuelles von der C'85 in Köln	15	08/85
	Btx Total (Internationale Funkausstellung)	8	10/85
	PCW-Computermesse in London	8	11/85
	Neues von der Commodore-Fachausstellung 1985	8	12/85
Recht	Die neue Abmahnmasche — Vorsicht bei Programmangeboten	8	05/85
	Die Ex-Knacker — wo sind sie geblieben?	27	08/85
	Interview mit Raubkopierern (Section 8)	28	08/85
	Schützer kontra Knacki's	23	08/85
	Raub-Talkshow	12	08/85
	Das Urheberrechtsgesetz und Gedanken zu seiner Anwendung	21	08/85
	Änderung des Urheberrechtsgesetzes	162	09/85

Buchbesprechungen

Stichwort	Titel	Seite	Ausgabe
Anfänger	Goldmann Computer Compact	87	03/85
	Basic-Wegweiser für den C 64	86	05/85
	Alles über den C 64, Sachbuchreihe, Band 1	115	06/85
	Lehrspielzeug Computer: C 64/VC 20	112	11/85
	C 64 Computerhandbuch	171	11/85
	Einführungskurs: Commodore 64	144	12/85
Anwendung	Dienstprogramme VC 20, C 64 und SX	86	05/85
	Spaß an Mathe mit dem Commodore 64	88	07/85
	Mathe für die Oberstufe mit dem C 64	88	07/85
	Mathematische Routinen VC 20, Elektrotechnik/ Elektronik	112	11/85
	Commodore 64-Listings, Band 2: Dateiverwaltung, Schule, Hobby	112	11/85
	Das Trainingsbuch zum Datamat	144	12/85
C 128	Bücher zum C 128	22	10/85
DFÜ	Das Mailbox-Jahrbuch: Nutz die Netze	112	11/85
Grafik	Grafik auf dem Commodore 64 (+Fehlert. 9/85)	86	05/85
	Einführung in CAD mit dem Commodore 64	128	06/85
	Grafik & Musik auf dem Commodore 64	88	07/85
	Verschiedene Grafikbücher zum C 64	115	08/85
Programmieren	Von Basic zu Assembler: Das Commodore-Buch, Band 4	115	06/85
	64 Intern	115	06/85
	Das Interface Age System-Handbuch zum C 64	115	06/85
	Das C 64 Buch, Band 5: Simons Basic Leitfaden	144	12/85
	Basicode	144	12/85
	Noch mehr Tips und Tricks zum 64er	144	12/85
Speichern	Das Kassettenbuch zum C 64 und VC 20	87	03/85
	Die Floppy 1541 (M&T)	88	07/85
Spiele	Rombachs C 64 Spielführer	87	03/85
	Commodore 64-Listings, Band 1, Spiele	112	11/85
	35 ausgesuchte Spiele für Ihren Commodore 64	171	1/85

64'er Extra

Stichwort	Titel	Seite	Ausgabe
Prozessor	Befehlssatz des 6502/6510-Prozessors	84	09/85
Grafik	Die Videochip-Register des C 64	92	10/85
Sound	Der SID-Chip, seine Register und Programmierung	92	11/85
Speicher	Die Speicherbelegung des C 64	96	12/85

Abenteuerlösungen

Stichwort	Titel	Seite	Ausgabe
Lösungen	Dallas-Quest Lösung	90	01/85
	Lösung The Hobbit	49	02/85
	Guncho Krill-Enchanter ist gelöst	44	03/85
	Infocom-Geheimnisse gelüftet?	49	05/85
	Des Rätsels Lösung: Amazon	145	06/85
	Activision-Adventures entschleiert (Mindshadow, Tracer Sanction)	36	12/85
	Eureka! — ich hab's!	37	12/85
	Lösungen zu Hitchhiker's Guide und Sorcerer	39	12/85

Spiele-Tests

Stichwort	Titel	Seite	Ausgabe
007	James Bond — A View to a Kill	156	09/85
Abenteuer	Abenteuerpaket I	48	08/85
	Amazon — das besondere Adventure	49	04/85
	Gordon Saga	48	02/85
	Shadowfire	146	09/85
	The Quest — mit C 64 auf Suche nach Drachen	47	01/85
Action	Hexenküche	50	07/85
	Impossible Mission	46	02/85
	Master of the Lamps	48	07/85
	Rescue on Fractalus	158	10/85
	Stellar 7	49	08/85
Construction	Mail Order Monsters	49	08/85
Set	Racing Destruction Set	50	06/85
Geschicklichkeit	Australopedicus Robustus	50	08/85
	Boulder Dash II	159	10/85
	Crystal Castles	50	07/85
	Gribbly's Day out	148	09/85
	Rock'n Bolt	48	08/85
	Thing on a Spring	159	10/85
	Tom + Zaga	48	01/85
Pseudo-	Roland's Rat Race	49	08/85
Adventures	Fourth Protocol und Frankie g.t.H.	162	11/85
Renner	Die Renner 1985: Meistverkaufte Spiele	34	12/85
Schach	Viermal Schachmatt: Verschiedene Schachprogramme	32	12/85
Simulation	Elite	148	09/85
	Jump Jet	148	09/85
	Super Huey Hubschraubersimulator	49	07/85
Sport	Boxspiele: Frank Bruno's B. + Barry McGuigan Champions. B.	49	12/85
	Handkantenschlag per Joystick: Karateka + Exploding Fist	165	11/85
	Nick Faldo Plays the Open (Golf)	159	10/85
	Rallye Speedway	49	07/85
	Slapshot (Eishockey)	50	07/85
	Summer Games II	146	09/85
	World Series Baseball	49	07/85
Diverses	New York City und Air Support	145	06/85

Hardware-Tips und Bauanleitungen

Stichwort	Titel	Seite	Ausgabe
Audio/Video	Besseres Monitorbild beim C 64	90	02/85
	Richtig verbunden — Video/Audio Kabel C 64 (+ Fehlerteufel 3/85)	22	02/85
	Mit 5 Mark zu neuen Dimensionen (Stereoanlage am C 64)	34	05/85
	Ein Monitor ist genug (RGB+Composite an C 128)	16	10/85
C 16	Alte Datasette am C 16	31	04/85
	Alter Joystick am C 16	35	05/85
Eingabegeräte	Der Hexer — Zusatztastatur für den MSE	48	10/85
EPROM	EPROMs im Expansion-Port	46	10/85
	EPROM-Trans — Die Super-Erweiterung	42	10/85
	Das 64'er EPROM-Programmiergerät, Teil 1	44	12/85
Floppy/Datasette	Diskettenlaufwerk 1541 selbst justiert	32	10/85
	Die Datasette streikt nie wieder (Anpassung des Tonkopfs)	34	10/85
IEC-Bus	Auf zu neuen Welten: IEC-Bus im Selbstbau (+Fehlerteufel 10/85)	44	07/85
Joystick	Joystick im Selbstbau	33	03/85
	Dauerfeuer-Adapter	46	08/85
RS232/V.24	Das 30-Mark-Interface (Selbstbau RS232)	29	03/85
	Genau betrachtet: Die RS232/V.24-Schnittstelle	80	05/85
VC 20	16 KByte-Erweiterung umschaltbar	20	02/85
	Der VC 20 steuert Super 8-Kamera	70	02/85
Diverses	Userport-Display	36	05/85
	Reset-Taster für alle Fälle (+Fehlert. 9/85)	130	06/85
	Aus eins mach vier (absturzfreie Betriebssystemumschaltung)	41	07/85

Hardware-Grundlagen

Stichwort	Titel	Seite	Ausgabe
C 16	C 16 — großer oder kleiner Bruder des C 64?	29	04/85
Computer	Was bringt der C 128?	28	11/85
Drucker	Welcher Drucker ist der Richtige? (Grundlagen)	15	05/85
	Hammerwerke — wie funktionieren Typenraddrucker	32	06/85
	Die Alternativen: Thermo-, Tintenstrahldrucker + Plotter	24	07/85
Eingabegeräte	Versteht Sie Ihr Computer? (Wie funktionieren Eingabegeräte)	44	09/85
Floppy	Floppy oder Datasette?	129	06/85
Monitore	Wie funktionieren sie, was ist beim Kauf zu beachten?	16	12/85
	Das Kabel zum Monitor: Welche Normen gibt es?	28	12/85
Peripherie	Grafikeingabegerät: Wie funktionieren sie?	30	06/85

Hardware-Tests

Stichwort	Titel	Seite	Ausgabe
80 Zeichen	Mit 80 fängt das Leben an (Test 80-Zeichen-Karten) (+Fehlert. 6/85)	17	04/85
Computer	Generationswechsel: Test C 16	16	01/85
	Plus und Minus beim Plus/4	14	02/85
	PC 128 — der Profi (Hardwaretest)	13	04/85
	Erster ausführlicher Test C 128 PC (Teil 1)	16	06/85
	Erster ausführlicher Test C 128 PC (Teil 2)	17	07/85
DFÜ	Marktübersicht Modems & Akustikkoppler	32	07/85
Drucker	Vergleich: Drucker unter 700 Mark (Tests und Marktübersicht)	18	05/85
	Tests und Marktübersicht Typenraddrucker	35	06/85
	Test: Brother EP 44	27	07/85
	Brother TC-600	118	08/85
	Riteman C+	133	09/85
	Panasonic KX-P1091	134	09/85
	Star SG 10C	132	09/85
	Melchers CP-80X — wie hätten Sie's denn gern?	25	10/85
	Geheimtip: Der RFI DP 165	24	10/85
	Epson GX 80 — einer für alle	26	10/85
	MPS 803 — ein Drucker für alle Gelegenheiten?	40	1/85
	Epson JX-80 das vielfarbige Druck-Genie	38	11/85
	Epson FX-85 neue Referenz	42	11/85
	SP 1000 VC — Superstar mit Haken	41	11/85
	Der NEC-P2 — das fernöstliche Wunder	159	12/85
	DMPG9 — eine solide Sache	162	12/85
Eingabegeräte	Der Bildschirm wird zur Leinwand (TechSketch-Lightpen)	21	04/85
	Das Doppelleben des Joystick-Ports: 10er-Tastaturen	50	09/85
	Joysticks: Test und Marktübersicht (+Fehlerteufel 12/85)	19	11/85
	Es geht auch anders: Lightpens und Trackballs	22	11/85
EPROMer	Frisch gebrannt ist halb gespeichert (EPROM-Programmiergeräte im Test)	39	07/85
	QuickByte II — das Kraftpaket	14	10/85
Floppy/Datasette	Schnell wie der Wind (Test: Speeddos, Turbo-Access)	22	04/85
	Turbo-Floppies, zweite Generation: Speeddos plus + Prologic DOS	28	10/85
	Das große Rennen: Schnelle Bandlaufwerke	37	10/85
	Professionelle Floppylaufwerke für den C 64 (IEC-Floppies)	30	10/85
	Gut gekauft ist halb gespeichert (Marktübersicht Disketten)	38	10/85
Grafik	Die Videowerkstatt (Digitizer-Test)	32	05/85
	Digitalbilder m.d. C 64: PrintTechnik Digitizer	24	01/85
Interface	Hardware-Interface ganz weich: Test EC 64	23	01/85
	Gute Connections — Übersicht Schnittstellen	21	03/85
	Card/Print +6 — Das Allround-Interface	20	03/85
	Das Wiesemann-Centronics-Interface	18	03/85
	Erst ein IEC-Bus öffnet Tür und Tor (+Fehlert.4/6-85)	24	03/85
Monitore	Eine klare Sache: Test Phönix-Monitor	28	04/85
	Marktübersicht: Monochrome Monitore	30	12/85
Musik	Die Stimme des Meisters: Test Voice Master	19	02/85
	Trommelwirbel: Test Digital Drums	45	08/85
	Die Musikhardware zum C 64	17	09/85
Oszilloskop	Der C 64 als Speicheroszilloskop	26	04/85
Roboter	Roboter selbst gebaut (Fischertechnik)	167	10/85
Scanner	So lernt Ihr Drucker lesen	30	06/85
Speicher	Speichertuning VC 20: Test 64 KByte Karte	26	01/85
Steuern	Flottes Türmchen: MEA-Interface	116	08/85

Kurse

Stichwort	Titel	Seite	Ausgabe
Assembler	Assembler ist keine Alchimie, Teil 5	142	01/85
	Assembler ist keine Alchimie, Teil 6	134	02/85
	Assembler ist keine Alchimie, Teil 7	124	03/85
	Assembler ist keine Alchimie, Teil 8	138	04/85
	Assembler ist keine Alchimie, Teil 9	138	05/85
	Assembler ist keine Alchimie, Teil 10	127	07/85
	Assembler ist keine Alchimie, Teil 11	126	08/85
	Assembler ist keine Alchimie, Teil 12	109	09/85
	Assembler ist keine Alchimie, Teil 13 (Schluß)	143	10/85
C 128	Entdeckungsreise duch den C 128	42	12/85
Comal	Comal — Eine Einführung, Teil 3	130	02/86
Effektives Programmieren	Müllabfuhr im Computer: Garbage Collection, Teil 1	122	01/85
	Stringprogrammierung in Maschinensprache, Teil 2	147	02/85
	Finden mit System, eine neuartige Suchmethode, Teil 3	148	03/85
	Sortieren mit dem Computer, Teil 1	148	04/85
	Sortieren mit dem Computer, Teil 2	159	05/85
	Sortieren mit dem Computer, Teil 3	124	06/85
	Sortieren mit dem Computer, Teil 4	138	08/85
	Sortieren mit dem Computer, Teil 5	124	09/85
	Sortieren mit dem Computer, Teil 6 (Schluß)	150	12/85
Extern	C 64 extern — Der Weg nach draußen, Teil 1	144	08/85
	C 64 extern — Der Weg nach draußen, Teil 2	122	09/85
	C 64 extern — Der Weg nach draußen, Teil 3 (Schluß)	129	10/85
Floppy	In die Geheimnisse der Floppy eingetaucht, Teil 4	148	01/85
	In die Geheimnise der Floppy eingetaucht, Teil 5	130	03/85
	In die Geheimnise der Floppy eingetaucht, Teil 6	145	05/85
	In die Geheimnisse der Floppy eingetaucht, Teil 7 (Schluß)	116	06/85
	Directory-Manipulationen I	140	06/85
Floppy	Directory-Manipulationen II	163	10/85
Grafik	Hires 3 — die Grafikerweiterung zum Grafikkurs, Teil 1	123	02/85
	Hires 3 — 15 neue Basic-Befehle, Teil 2	136	03/85
	Hires 3 — Grafikkurs-Anwendung, Teil 3 (Schluß)	152	08/85
	Sprites ohne Geheimnisse	40	08/85
	Streifzüge durch die Grafikwelt, Teil 1	106	09/85
	Streifzüge durch die Grafikwelt, Teil 2	149	11/85
Logeleien	Logeleien, Teil 1	143	07/85
	Logeleien, Teil 2	136	08/85
	Logeleien, Teil 3 (Schluß)	115	09/85
Musik	Dem Klang auf der Spur, Teil 2	136	01/85
	Dem Klang auf der Spur, Teil 3	152	02/85
	Dem Klang auf der Spur, Teil 4	131	04/85
	Dem Klang auf der Spur, Teil 5	152	05/85
	Dem Klang auf der Spur, Teil 7	132	07/85
	Dem Klang auf der Spur, Teil 8	133	08/85
	Dem Klang auf der Spur, Teil 9	126	10/85
	Dem Klang auf der Spur, Teil 10 (Schluß)	157	11/85
Speicher	Memory Map mit Wandervorschlägen, Teil 3	126	01/85
	Memory Map mit Wandervorschlägen, Teil 4	150	02/85
	Memory Map mit Wandervorschlägen, Teil 5	144	03/85
	Memory Map mit Wandervorschlägen, Teil 6	144	04/85
	Memory Map mit Wandervorschlägen, Teil 7	120	06/85
	Memory Map mit Wandervorschlägen, Teil 8	140	07/85
	Memory Map mit Wandervorschlägen, Teil 9	129	08/85
	Memory Map mit Wandervorschlägen, Teil 10	112	09/85
	Memory Map mit Wandervorschlägen, Teil 11	133	10/85
	Memory Map mit Wandervorschlägen, Teil 12	145	11/85

64er-online.de
64er-online.net

Stichwort	Titel	Seite	Ausgabe
	Memory Map mit Wandervorschlägen, Teil 13	146	12/85
Sprachen	Basic ist out — es lebe Forth	43	01/85
VC 20	Der gläserne VC 20, Teil 4	130	01/85
	Der gläserne VC 20, Teil 5	141	02/85
	Der gläserne VC 20, Teil 6 (Schluß)	155	03/85

Software-Tips

Stichwort	Titel	Seite	Ausgabe
C 128	Erste Fragen und Antworten zum C 128	14	09/85
	Fragen und Antworten zum 128er	20	10/85
	Fragen und Antworten zum 128er	40	12/85
Drucker	Der MPS 802 lernt Deutsch	30	05/85
	Centronics-Interface für jeden Bedarf	78	07/85
Textverarbeitung	Software Corner — professionelle Programme richtig eingesetzt (Vizawrite-Tips)	174	12/85
Tips & Tricks	Autoboot beim C 64	86	03/85
	Verbindungsfreundlich (Parallelschnittstelle des VC 20)	91	03/85
	Undefinierte Opcodes des 6502	84	03/85
	Durch POKEs zum Erfolg (Spiele-POKEs)	83	03/85
	Tips und Erweiterungen zu Hi-Eddi und Simons Basic	88	03/85
	Hardcopy mit einer Zeile (MPS 801)	82	04/85
	VC 20-Programme schützen	83	04/85
	64-Tastaturänderung	63	04/85
	Basic-Befehle im Griff	79	05/85
	Durch POKEs zum Erfolg: Spiele-POKEs	78	06/85
	Formatierte Eingabe	148	06/85
	Hi-Text (Text in Hires)	70	08/85
	Verbotene Variablen	66	09/85
	Verschiedene Routinen für Anfänger und Profis (+Fehlerteufel 12/85)	88	11/85
	Der Trick mit dem Joystick (Joystickabfrage)	24	11/85
	Verschiedene Tips für Anfänger und Fortgeschrittene	106	12/85

Software-Grundlagen

Stichwort	Titel	Seite	Ausgabe
Assembler	Assembler? Assembler! (Einführung)	32	01/85
	Assembler-Bedienung leicht gemacht, Teil 1	169	12/85
Compiler	So arbeiten Compiler	39	02/85
DFÜ	Ein modernes Abenteuer — Mailboxen in Deutschland	43	04/85
	Der erste Kontakt mit DFÜ	40	06/85
	Die Netze der Post: Btx, Datex-P, Telebox	46	06/85
	DFÜ — Was ist das?	44	06/85
	Mailbox für Anfänger	30	07/85
Datei	Die wichtigsten Begriffe der Dateiverwaltung	42	05/85
	Dateiverwaltung ist nicht gleich Datenbank	44	05/85
	Dateiverwaltung: Was Sie beim Kauf beachten sollten	40	05/85
Drucker	Hardcopy leicht gemacht (wie programmiert man Hardcopies)	34	09/85
EPROM	Wie sage ich es meinem EPROM? (EPROM-Grundlagen)	35	07/85
Funktionen	Funktionen für Anfänger	164	05/85
Lernen	Besser lernen mit dem Computer	166	10/85
Musik	Klangprogrammierung ohne Ballast	19	09/85
Spiele	Taktik- und Strategiespiele	46	03/85
	Play by Mail und Play by Modem	153	09/85
Sprachen	Sprachen für Computer	47	04/85
	Sprachen für Computer, Teil 2	46	05/85
Textverarbeitung	Von der Schreibmaschine zum Textsystem	34	03/85

Listings zum Abtippen

Stichwort	Titel	Seite	Ausgabe
Anwendung	Der C 64 als Handballtrainer (AdM)	52	01/85
	Familienplanung (AdM)	52	02/55
	Ligatab — ohne Organisation kein Tor (LdM)	50	03/85
	Gut Ziel mit dem C 64 — Schützenvereinsergebnisse (AdM)	52	03/85
	Weißt du, wieviel Sternlein stehen (Sternkarte) (AdM) (+Fehlert. 6/85)	52	05/85
	Haushaltsbuchführung (AdM)	52	07/85
	Netzwerkanalyse: Ein Programm für Hobbyelektroniker (AdM)	52	08/85
	Prüfungsfragen (AdM)	52	09/85
	Fit in Latein mit dem C 64 (AdM)	52	10/85
	Lyrik-Maschine (AdM)	52	11/85
	Hypra-Platos (LdM)	50	11/85
	Der Chemie-Assistent (AdM)	52	12/85
	SMON Teil 3: Ohne gutes Werkz. geht es nicht	69	01/85
	SMON Teil 4 (+Fehlerteufel 4/85)	72	02/85
	SMON Teil 5 (+Fehlerteufel 5/85)	64	04/85
	Hypra-Ass (LdM)	51	07/85
	Neues vom SMON (+Fehlerteufel 11/85)	87	10/85
	Reassembler zu Hypra-Ass (+Fehlerteufel 12/85)	97	11/85
	Ergänzungen zu Hypra-Ass (bedingte Verzweigungen)	96	11/85
	Tips & Tricks zum SMON (inklusive Diskmonitor)	100	12/85
Basic-Erweiterung	Befehlserweiterung C 64: Bildschirmsteuerung und Masken	80	04/85
	xBasic 64: eine Super-Basic-Erweiterung (LdM) (+Fehlerteufel 5/85)	52	04/85
Bildschirmseite	Auflösung Wettbewerb Bildschirmseite: Drei Top-Programme	158	09/85
DFÜ	Terminalprogramm der Spitzenklasse (+Fehlerteufel 10/85)	149	07/85
Datei	SMU — Der Maskengenerator (LdM)	50	12/85
Drucker	Print-List (formatierte Listings)	79	04/85
	Hi-Eddi-Druckerroutinen	69	06/85
	C 64 Schreiberling — Drucken wie gemalt	54	10/85
	Koalabilder-Farbhardcopy auf Epson JX-80	39	11/85
Einzeiler	Die nächsten 14 aus d. Einzeilerwettbewerb	157	01/85
	11 neue Einzeiler! (+Fehlerteufel 5/9/85)	153	04/85
Floppy	Hypra-Load mal 4 (+Fehlerteufel 3/85)	82	01/85
	Neues vom Hypra-Load: Hypra-Perfekt	75	04/85
	Diskettenmonitor	83	08/85
	Disk-Designer	70	09/85
	Herzoperation (Hypra-Load + Hypra-Ass + DOS5.1 + Centronics)	104	11/85
Grafik	Vier Pseudo-VICs mit 32 Sprites	76	01/85
	Hi-Eddi: Zeichen- und Malproggramm (LdM)	50	01/85
	Als die Bilder laufen lernten (Pseudo-Scroll)	88	02/85
	Elektrotechnisches Zeichnen mit dem VC 20	71	03/85
	Supergrafik III (3D-Grafiken mit dem VC 20)	73	04/85
	Funktionen im Netz (3D-Grafik)	69	04/85
	Window 64 — Fenstertechnik für den Commodore	87	04/85
	Mini-Grafik VC 20, Grafikhilfe	69	05/85
	Trickfilm mit dem C 64: Bewegte 3D-Grafik (LdM) (+Fehlerteufel 6/85)	51	05/85
	Kurvenplotten mit Hardcopy auf dem C 16	68	06/85
	Doppelte Grafikauflösung für C 128	33	11/85
	Bilder aus einer anderen Dimension (Apfelmännchen)	80	11/85
Intelligenz	VIC — das intelligente Programm (Wettbewerbssieger)	173	05/85
Musik	Sound Machine (+Fehlerteufel 10/85)	23	09/85
	Sound Master (Basic-Erweiterung)	31	09/85
Schach	Schachmeister erweitert	68	04/85
Spiele	Das Grab des Pharaoh (LdM) (+Fehlerteufel 3/85)	51	02/85
	Q - Bert (VC 20)	78	02/85
	Gehirntraining mit Super Memory	81	02/85
	6510 — Die Suche nach der Prozessor	70	05/85
	Samurai (Strategiespiel)	72	06/85
	Schach dem C 64: Schachprogramm zum Abtippen	72	08/85
	Spielen auf zwei Bildschirmen: Zeichensatzscrolling (LdM)	51	09/85
	Pac-Man unter der Lupe	76	10/85
	Block Out	84	11/85
	Seekrieg per Telefon (Schiffe versenken per Modem)	82	12/85
Spielehilfe	Die Scroll-Maschine — D. Fenster zur Spielewelt (LdM) (+Fehlert. 11/85)	52	06/85
Sprachen	Tiny Forth Compiler (LdM) (+Fehlert. 9/85)	51	08/85
Textverarbeitung	Hypra-Text (LdM) (+Fehlerteufel 11/85)	50	10/85
	Drucksache — Hypra-Text, Teil 2	71	11/85
Tips & Tricks	Große Buchstaben	89	01/85
	Restore für Unterprogramme	90	01/85
	Parameterübergabe an Maschinenspracheprogramme	88	01/85
	Cursorsteuerung leicht gemacht	86	02/85
	Maschinenspracheprogramme auf Disk speichern	91	02/85
	Basic-Zeilen genau betrachtet	87	02/85
	RAM-Floppy	92	02/85
	22 Read Error — Theorie und Praxis	41	03/85
	Floppy-Lister (+Fehlerteufel 4/85)	82	03/85
	Longscreen beim VC 20	83	05/85
	C 16: Help und Trace verbessert	84	05/85
	Ordnung ist das halbe Leben (Directory-Sorter)	77	05/85
	Dokumentationshilfe, Cross-Referenz-Liste C 64 (Wettbewerb)	155	06/85
	Prost mit dem C 64: Gerätesteuerung über Userport (+Fehlerteufel 9/85)	76	06/85
	Fenster-Befehle für den C 16	84	07/85
	Elektronische Merkzettel	83	07/85
	File-Compactor	82	07/85
	REM-Killer (+Fehlerteufel 9/85)	75	07/85
	Basic-Start-Generator	74	07/85
	Komfortable Ein-/Ausgaberoutine	77	07/85
	Bildschirmmasken leicht erstellt	86	08/85
	Der Bitmap-Compander (HiRes-Bilder komprimieren)	81	08/85
	Hypra-Save	79	08/85
	'Procedure' — oder der C 64 kann lernen	78	08/85
	Aufgewickelt — Listingscrolling für VC 20	63	09/85
	Programmgenerator für den C 64	86	10/85
	Cross-Ref optimiert	83	10/85
	Spieletrainer: Spritekill	86	11/85
	Tipp-Utility	99	12/85
	Der EPROM-Automat (wie man Module macht)	90	12/85
	80-Zeichen-Grafik für den C 128	78	12/85
	Hyper Screen (Sprites auf dem Bildschirmrand)	76	12/85
Transfer	Der C 64 als PET: PET-Simulator	87	01/85
Unterprogramme	Formatierte Eingabe	156	01/85
Witz	Notlandung (Das lustigste Programm)	156	02/85
	Epson bedruckt Ostereier (AdM) (+Fehlerteufel 5/85)	50	04/85

Software-Tests

Stichwort	Titel	Seite	Ausgabe
Assembler	Assembler im Test Teil 1	34	01/85
	Assembler im Test, Teil 2	30	02/85
Basic-Compiler	Basic-Compiler im Test (+Fehlerteufel 5/85)	34	02/86
Basic-Erweiterung	GBasic — Alles drin	28	01/85
	Aztec Basic — von jedem etwas	42	04/85
	Macro-Basic: Die Unterprogramm-Bibliothek	137	06/85
	Darf es etwas mehr sein? — Test Business-Basic	120	08/85
	Das Intellectool	138	09/85
	Formel 64: Das Multitalent	158	12/85
Compiler	Basic 64 — ein vielseitiger Basiccompiler	36	04/85
DFÜ	Terminal 64 — Schwer auf Draht	24	02/85
	Terminalprogramme: Übersicht	42	06/85
Datei	Vergleichstest — 7 Dateiverwaltungen auf einen Blick	118	07/85
	Aufgeräumt mit Mainfile II	157	10/85
Grafik	Ich glaub, mein Drucker pfeift (Test: Printshop)	34	04/85
	Malkasten ade! (Test: Blazing Paddles)	40	04/85
	Malen auf dem Bildschirm (Malprogramme)	34	08/85
	Grafikprogramme auf einen Blick: Marktübersicht	38	08/85
	Vergleichstest: Grafik-Erweiterungen	37	09/85
Lernen	Softlearning — die weiche Welle des Lernens	40	01/85
	Nachhilfe (Übersicht Lernsoftware)	26	02/85
	Vokabeltraining mit dem Computer	39	03/85
	Marktübersicht: Lernsoftware	168	10/85
Musik	Musik für den C 64: Übersicht Musiksoftware	26	09/85
	The Music System — Zwei auf einen Schlag	164	12/85
Sprachen	Logo — die Sprache für Einsteiger	135	05/85
	Der Ada Trainingskurs auf dem C 64	129	05/85
	Promal — die neue Sprache für Profis?	124	07/85
	Forth-wärts mit M&T-Forth 64	126	07/85
	Was leistet Pilot?	121	08/85
	Pascal für Profis (Profi-Pascal)	122	08/85
	Super-Forth 64	144	09/85
	C — die professionelle Programmiersprache für den C 64	140	09/85
	Basic 7.0 — Das Superbasic des C 128	18	10/85
	Comal 80 — die universelle Programmiersprache	151	10/85
	Turbo-Pascal auf dem C 128	30	11/85
Textverarbeitung	Homeword - Textverarbeitung zu Hause	36	03/85
	Totl-Text — Flexibilität ist Trumpf	38	03/85
	Texte gut im Griff (Übersicht Textverarbeitung) (+Fehlerteufel 5/85)	38	04/85
	Protext — Textprofi mit 80 Zeichen	133	05/85
	Textomat Plus kontra Vizawrite	132	06/85
	Der Preishammer (Test: StarTexter)	135	09/85
	Paperclip — ausdrücklich gut	44	11/85

So machen's andere

Stichwort	Titel	Seite	Ausgabe
Lernen	Gelungener Einstieg (Informatik-Unterricht)	159	04/85
Semmeln	Semmelservice mit dem C 64	147	06/85
Sport	Commodore Sportservice: Heimcomputer zur Turnierauswertung	157	07/85
Hilfe	Computer für Behinderte	182	12/85

Auch die bisher erschienenen Sonderhefte können Sie jetzt direkt bestellen:

SONDERHEFT 01/84: TIPS & TRICKS
Unentbehrliche Anwendungslistings für C 64 und VC 20.

SONDERHEFT 02/85: ABENTEUERSPIELE
Fesselnde Adventures mit zahlreichen Lösungen und einem Programmierkurs.

SONDERHEFT 03/85: SPIELE
Heiße Listings für Spiele-Fans und eine große Marktübersicht.

SONDERHEFT 04/85: GRAFIK & DRUCKER
Von der 3D-Darstellung bis zur Hardcopy-Routine.

SONDERHEFT 05/85: FLOPPY/DATASETTE
Soft-Tools zum komfortablen und noch schnelleren Betrieb von Floppy und Datasette.

SONDERHEFT 06/85: AUSGEWÄHLTE SUPER-LISTINGS
Top-Themen aus 64'er bringt eine Auswahl der besten 64'er Programme.

SONDERHEFT 07/85: ANWENDUNGEN/DFÜ
Leistungsfähige Programme für professionelle Anwendungen und Datenfernübertragung.

Am besten gleich mitbestellen: Die 64'er-Sammelbox

Für alle Leser, die »64'er« regelmäßig kaufen, sammeln oder im Abonnement beziehen, gibt es jetzt ein interessantes Service-Angebot: die 64'er-Sammelbox!

Mit dieser Sammelbox bringen Sie nicht nur Ordnung in Ihre wertvollen Hefte, sondern schaffen sich gleichzeitig ein interessantes und attraktives Nachschlagewerk.

Übrigens: Die Sammelbox ist nicht nur ein praktisches Aufbewahrungsmittel: Sie eignet sich auch hervorragend als Geschenk für Freunde und Bekannte zu vielen Anlässen.



64er-online.de

Nach Eingabe der entsprechenden Daten erscheint wieder das Eingabemenü. Die Daten der zu vergleichenden Splines können nun entweder über die Tastatur, das Bandgerät oder die Diskettenstation eingegeben werden.

Nach Eingabe der in der jeweils gewählten Option gefragten Angaben gelangt man ins Ausgabemenü (Bild 3).

NEW (1)
zurück ins Eingabemenü.

TAB (2)
Ausgabe der ermittelten Splinefunktionen auf dem Bildschirm; Ausgabe der x,y-Werte der Stützstellen sowie der errechneten Interpolationswerte in Tabellenform.

COPY (3)
Ausgabe der entsprechenden Tabelle (TAB (2)) auf einem Drucker MPS 801.

GRAPH (4)
Grafische Darstellung im HiRes-Mode (Simons Basic). Wurde zuvor eine vollständige Grafik von Diskette oder Band geladen, so erfolgt nach Wahl dieser Option direkt die grafische Ausgabe auf dem Bildschirm.

Für die Darstellung eines Datensatzes, der über die Optionen KEY oder D-FILE eingegeben wurde, sind einige Angaben zur Beschriftung der Koordinatenachsen (Einheiten) und der Achseneinteilung erforderlich. Die Einteilung kann sowohl manuell (MANU) als auch automatisch (AUTO) erfolgen. Bei manueller Eingabe der Achseneinteilung können lineare oder logarithmische Maßstäbe für Ordinate (y-Achse) und Abszisse (x-Achse) gewählt werden. Des weiteren erwartet das Programm Angaben über die Begrenzung (MAXIMALWERT, MINIMALWERT) und Einteilung (STUFUNG) des Diagramms, jeweils für X-ACHSE und Y-ACHSE.

OLD (5)
Nach Wahl dieses Menüpunktes erscheint die zuletzt erstellte Grafik wieder auf dem Bilschirm.

COPY (6)
Erzeugt Hardcopy der Grafik auf einem Drucker MPS 801.

TAPE (7), DISK (8)
Nach Wahl einer dieser Optionen wird die vollständige Grafik auf Band oder Diskette als Datensatz abgelegt.

Abschließend noch zwei Hinweise zur allgemeinen Bedienung des Programms. Unterbricht der Programmablauf zur Darstellung einer Grafik oder Tabelle, so wird es nach Drücken der SPACE-Taste fortgesetzt. Bei nahezu allen Eingaben ist es möglich, durch Drücken der »←«-Taste einen Rückschritt zu erzeugen, um Korrekturen durchzuführen. Entsprechend dem speziellen Format bei der Eingabe der Stützstellen bedeutet hier »←,0« einen Rückschritt.

Beispiele

Bild 4 zeigt den Wertverlust zweier Mittelklasse-Pkw im Vergleich. Die Wertepaare (Jahre, DM) wurden einer ADAC-Zeitschrift entnommen. Die obere Kurve repräsentiert den Kadett L 1.3, während die untere Kurve den Preisverfall des Peugeot 305 GL zeigt.

Als Beispiel eines durchgeführten Programmlaufs in der Option FKT-SPL zeigen die Bilder 5 bis 7 den Vergleich einer durch Splines erzeugten Parabel mit dem Graphen der Funktion $f(x)=x^2$. Für die Spline-Berechnung wurden 10 Interpolationsschritte und die folgenden Stützstellen gewählt:

x	-3	-2	-1	0	1	2	3
y	9	4	1	0	1	4	9

Das Ergebnis zeigt Bild 5. Die Funktionsdaten für die Darstellung des Funktionsgraphen $f(x)=x^2$ (Bild 6) waren:

ANFANGS X-WERT : -3
END X-WERT : 3
ANZAHL ZWISCHENWERTE : 20

Die Überlagerung der beiden Kurven in Bild 7 zeigt kaum erkennbare Abweichungen und beweist damit die Leistungsfähigkeit des Spline-Verfahrens.

(M. Buhtz/ah)

```
10 REM ***********************
20 REM *       SPLINE        *
30 REM *                     *
40 REM *         BY          *
50 REM *                     *
60 REM *       M.BUHTZ       *
70 REM *                     *
80 REM *  TEL.: 0281/22431   *
90 REM ***********************
100 RUN110
110 REM  *** NATUERLICHE SPLINEFUNKTION ***
120 :
130 REM EINGABEMENUE
140 :
150 PRINT"■■                   EINGABEMENUE
 ■"
160 PRINT:PRINT:PRINT:PRINTSPC(15)"KEY      (1)"
170 PRINT:PRINT:PRINTSPC(15)"TAPE       "
180 PRINT:PRINTSPC(15)"GRAPHIC (2)"
190 PRINTSPC(15)"D-FILE  (3)"
200 PRINT:PRINT:PRINTSPC(15)"DISK       "
210 PRINT:PRINTSPC(15)"GRAPHIC (4)"
220 PRINTSPC(15)"D-FILE  (5)"
230 PRINT:PRINT:PRINTSPC(15)"FKT-SPL (6)"
240 GETA$:A=VAL(A$):ON A GOTO 380,3560,4550,3570,4
560,4930:GOTO240
250 :
260 REM AUSGABEMENUE
270 :
280 PRINT"■■                 AUSGABEMENUE
 ■"
290 PRINT:PRINT:PRINTSPC(12)"NEW       (1)"
300 PRINT:PRINT:PRINT:PRINTSPC(12)"TAB       (2)"
310 PRINT:PRINTSPC(14)"COPY   (3)"
320 PRINT:PRINT:PRINT:PRINTSPC(12)"GRAPH     (4)"
330 PRINT:PRINTSPC(14)"OLD    (5)"
340 PRINT:PRINTSPC(14)"COPY   (6)"
350 PRINT:PRINTSPC(14)"TAPE   (7)"
360 PRINT:PRINTSPC(14)"DISK   (8)"
370 GETA$:A=VAL(A$):ON A GOTO  100,930,3380,1230,2
570,2540,3890,3900:GOTO370
380 :
390 REM EINGABE KEY
400 :
410 PRINT"■■                 EINGABE KEY
 ■"
420 PRINT:PRINT:INPUT"  ANZAHL DER KURVEN";AK$
430 IFAK$="←"THEN150
440 AK=VAL(AK$)
450 IFAK=0THEN420
460 PRINT:PRINT:INPUT"  ANZAHL DER INT.POL.SCHRITT
E";SW$
470 IFSW$="←"THEN420
480 SW=VAL(SW$)
490 IFSW=0THEN460
500 PRINT:PRINT
510 FORI=1TOAK
520 PRINT:PRINT"  ■KURVE"I"■"
530 PRINT:PRINT:INPUT"  ANZAHL DER STUETZSTELLEN";
N1$(I)
540 IFN1$(I)="←"THEN460
550 N1(I)=VAL(N1$(I))
560 IFN1(I)>=3THEN590
570 PRINT:PRINT:PRINT"  FEHLER!"
580 PRINT:PRINT"  MINIMALE ANZAHL STUETZSTELLEN =3
!":GOTO530
590 IFN1(I)>N1THENN1=N1(I)
600 NEXTI
610 PRINT:PRINT:PRINT"  EINGABE O.K. ?  (J/N)"
620 GETA$
630 IFA$="N"THEN410
640 IFA$="J"THEN660
650 GOTO620
660 N1=N1+1
670 DIMX(AK,N1),Y(AK,N1),SW(AK,N1)
680 DIMA(N1),C(N1),B(N1),D(N1),H(N1),M(N1,N1),V(N1
),O(N1),P(N1),Q(N1)
690 FORI=1TOAK
700 PRINT"■■              EINGABE STUETZSTELLEN
 ■"
710 PRINT:PRINT:PRINT"  ■KURVE"I"■"
720 FORJ=1TON1(I)
730 IFJ<=0THEN690
740 PRINT"■■  X("J"), Y("J")"
750 INPUT"■  ";X$(I,J),Y(I,J)
760 IFX$(I,J)="←"THENJ=J-2:NEXTJ
770 X(I,J)=VAL(X$(I,J))
780 DE=SW*(N1(I)-1)+N1(I)
790 IFDE>DITHENDI=DE
800 NEXTJ
810 GOSUB4160
820 FORJ=2TON1(I)
830 SW(I,J-1)=(X(I,J)-X(I,J-1))/SW
```

Listing. Das Programm »Spline« läuft nur mit Simons Basic

```
840 NEXTJ
850 PRINT:PRINT:PRINT"  ABSPEICHERN ?  (J/N)"
860 GETA$:IFA$="N"THEN890
870 IFA$="J"THEN4320
880 GOTO860
890 NEXTI
900 DIMZ(DI+3),T(DI+3)
910 GOTO260
920 :
930 REM TABELLE
940 :
950 :
960 FORK=1TOAK:G=0:N=N1(K):E6=1
970 GOSUB4100:GOSUB2610:J=0
980 PRINT"■■                 KURVE"K"■
    ■"
990 FORL=1TON1(K):PRINT
1000 A1=INT(A(L)*100+.5)/100:B1=INT(B(L)*100+.5)/1
00:C1=INT(C(L)*100+.5)/100
1010 D1=INT(D(L)*100+.5)/100
1020 PRINT"SPL■"L"■: Y=■"A1"■■+■"B1"■■*X+■"C1"■
*X↑2+■"D1"■■*X↑3"
1030 NEXTL
1040 IFPEEK(203)<>60THEN1040
1050 PRINT"■■ KURVE"K"■        X              Y
    ■"
1060 FORI=1TO21
1070 IFI+J>GTHEN1160
1080 ZA=ABS(Z(I+J)):TC=ABS(T(I+J))
1090 IFZA<1THENZA=0.1
1100 IFTC<1THENTC=0.1
1110 TA=INT(LOG(ZA)/LOG(10))+1:TB=INT(LOG(TC)/LOG(
10))+1
1120 PRINTTAB(13-TA)Z(I+J),TAB(28-TB)T(I+J)
1130 NEXTI
1140 IFPEEK(203)<>60THEN1140
1150 J=J+21:GOTO1050
1160 IFPEEK(203)<>60THEN1160
1170 IFE5= 2THEN1190
1180 NEXTK
1190 J=0:I=0:IFE5=1THEN5120
1200 IFE5=2THENE5=1:E6=0
1210 GOTO260
1220 :
1230 REM GRAPHIC
1240 :
1250 IFD<>0THENTX=NX:TY=NY:GOSUB4100:GOTO1920
1260 E2=0:E3=0
1270 PRINT"■■      ACHSENEINTEILUNG UND EINHEITEN
   ■"
1280 PRINT:PRINT:INPUT"        EINHEIT X-ACHSE";T1$
1290 IFT1$="←"THEN280
1300 PRINT:PRINT:INPUT"        EINHEIT Y-ACHSE";T2$
1310 IFT2$="←"THEN1270
1320 PRINT:PRINT:PRINT:PRINTSPC(12)"■ EINTEILUNG ■
"
1330 PRINT:PRINT:PRINT:PRINTSPC(14)"MANU   (1)"
1340 PRINT:PRINT:PRINT:PRINTSPC(14)"AUTO   (2)"
1350 GETA$:IFA$="←"THEN1270
1360 A=VAL(A$):IFA=2THENE1=1:GOTO1840
1370 IFA=1THENE1=0:GOTO1390
1380 GOTO1350
1390 PRINT:PRINT:PRINT"■                   ■ X-ACHSE ■"
1400 PRINT:PRINT:PRINTSPC(12)"LINEAR     (1)"
1410 PRINT:PRINTSPC(12)"LOGARITH.  (2)"
1420 GETA$:IFA$="←"THEN1320
1430 A=VAL(A$):ONAGOTO1440,1560:GOTO1420
1440 INPUT"■■      MAXIMALWERT";MX$
1450 IFMX$="←"THEN1390
1460 MX=VAL(MX$)
1470 INPUT"■■      MINIMALWERT";NX$
1480 IFNX$="←"THEN1670
1490 NX=VAL(NX$)
1500 IFNX>MXTHENPRINT:PRINT:PRINT"        FEHLER !"
:GOTO1440
1510 V=296/(MX-NX):TX=NX
1520 INPUT"■■      STUFUNG";SX$
1530 IFSX$="←"THEN1470
1540 SX=VAL(SX$)
1550 GOTO1610
1560 INPUT"■■■      MAXIMALWERT";MX$
1570 IFMX$="←"THEN1390
1580 MX=VAL(MX$)
1590 MX=LOG(MX)/LOG(10):SX=1:NX=0:TX=0:V=296/MX
1600 E2=1
1610 PRINT:PRINT:PRINT"■                   ■ Y-ACHSE "
1620 PRINT:PRINT:PRINTSPC(12)"LINEAR     (1)"
1630 PRINT:PRINTSPC(12)"LOGARITH.  (2)"
1640 GETA$:A=VAL(A$):ONAGOTO1670,1790:GOTO1640
1650 GETA$:IFA$="←"THEN1390
1660 A=VAL(A$):ONAGOTO1670,1790:GOTO1640
1670 INPUT"■■      MAXIMALWERT";MY$
1680 IFMY$="←"THEN1610
1690 MY=VAL(MY$)
1700 INPUT"■■      MINIMALWERT";NY$
1710 IFNY$="←"THEN1670
1720 NY=VAL(NY$)
1730 IFNY>=MYTHENPRINT:PRINT:PRINT"        FEHLER !
":GOTO1670
1740 W=176/(MY-NY):TY=NY
1750 INPUT"■■      STUFUNG";SY$
1760 IFSY$="←"THEN1700
1770 SY=VAL(SY$)
1780 GOTO1840
1790 INPUT"■■■      MAXIMALWERT";MY$
1800 IFMY$="←"THEN1390
1810 MY=VAL(MY$)
1820 MY=LOG(MY)/LOG(10):SY=1:NY=0:TY=0:W=176/MY
1830 E3=1
1840 PRINT:PRINT:PRINT"  EINGABE O.K. ?  (J/N)"
1850 GETA$
1860 IFA$="N"THEN1270
1870 IFA$="J"THEN1900
1880 :
1890 GOTO1850
1900 GOSUB4100
1910 :
1920 FORK=1TOAK:G=0:N=N1(K)
1930 :
1940 GOSUB2610
1950 :
1960 IFE1=0THEN2060
1970 MY=INT((MY+MY/10)*100+.5)/100
1980 NY=INT((NY-NY/10)*100+.5)/100
1990 W=INT((176/(MY-NY))*100+.5)/100:TY=NY
2000 SY=INT((MY-NY)*100+.5)/500
2010 MX=X(K,N):NX=X(K,1)
2020 V=INT((296/(MX-NX))*100+.5)/100:TX=NX
2030 SX=INT((MX-NX)*100+.5)/500
2040 E1=0
2050 :
2060 IFK>1THEN2340
2070 HIRES0,14
2080 LINE0,0,320,0,1
2090 LINE320,0,320,200,1
2100 LINE320,200,0,200,1
2110 LINE0,200,0,0,1
2120 FORJ= 0TO320STEPSX*V
2130 FORL=184TO0STEP-4
2140 PLOTJ,L,1
2150 NEXTL
2160 IFJ=0THEN2200
2170 T$=STR$(INT(TX*100+.5)/100)
2180 IFE2=1THENT$=STR$(INT(10↑TX*100+.5)/100 )
2190 TEXTJ-16,188,T$,1,1,8
2200 TX=TX+SX
2210 NEXTJ
2220 TEXTJ-SX*V*1.8,176,T1$,1,1,8
2230 FORJ=200TO0STEP-SY*W
2240 FORL= 0TO320STEP4
2250 PLOTL,J,1
2260 NEXTL
2270 IFJ=200THEN2310
2280 T$=STR$(INT(TY*100+.5)/100)
2290 IFE3=1THENT$=STR$(INT(10↑TY*100+.5)/100)
2300 TEXT0,J+2,T$,1,1,8
2310 TY=TY+SY
2320 NEXTJ
2330 TEXT8,J+SY*W*1.5,T2$,1,1,8
2340 IFE3=0THEN2390
2350 FORJ=1TOG
2360 IFT(J)<1THENT(J)=1
2370 T(J)=LOG(T(J))/LOG(10)
2380 NEXTJ
2390 IFE2=0THEN2440
2400 FORJ=1TOG
2410 IFZ(J)<1THENZ(J)=1
2420 Z(J)=LOG(Z(J))/LOG(10)
2430 NEXTJ
2440 FORI=1TOG-1
2450 Z1=(Z(I)-NX)*V   :Z2=(Z(I+1)-NX)*V
2460 T1=200- (T(I)-NY)*W :T2=200-(T(I+1)-NY)*W
2470 IFZ1<0ORZ2<0ORT1<0ORT2<0ORZ1>320ORZ2>320ORT1>
200ORT2>200THEN2490
2480 LINEZ1,T1,Z2,T2,1
2490 NEXTI
2500 IFE5=2THENE5=1:GOTO2570
2510 NEXTK
2520 IFE5= 1THEN5120
2530 GOTO2570
2540 PRINT"■":PRINT:PRINT:PRINT:PRINT:PRINTSPC(9)
"DRUCKER EINGESCHALTET ?"
2550 IFPEEK(203)<>60THEN2550
2560 CSET2:COPY
2570 CSET2
2580 IFPEEK(203)<>60THEN2580
```

Listing »Spline« (Fortsetzung)

64ER ONLINE

64er-online.de 64er-online.net

```
2590 CSET0:GOTO260
2600 :
2610 REM BERECHNUNG DER INT.POL.STELLEN
2620 :
2630 GOSUB2830
2640 :
2650 FORI=1TON-1
2660 X=X(K,I):Y=Y(K,I):GOSUB3280
2670 IFE1=1THENNY=Y(K,1)
2680 FORX=X(K,I)+SW(K,I)TOX(K,I+1) STEPSW(K,I)
2690 X=INT(X*100+.5)/100
2700 IFX=X(K,I+1)THEN 2750
2710 Y=A(I)+B(I)*(X-X(K,I))+C(I)*(X-X(K,I))↑2+D(I
)*(X-X(K,I))↑3
2720 :
2730 GOSUB3280
2740 :
2750 NEXTX
2760 NEXTI
2770 X=X(K,N):Y=Y(K,N)
2780 :
2790 GOSUB3280
2800 :
2810 RETURN
2820 :
2830 REM KOEFFIZIENTENBERECHNUNG
2840 :
2850 FORI=1TON
2860 A(I)=Y(K,I):NEXTI
2870 C(1)=0:C(N)=0
2880 FORI=1TON-1
2890 H(I)=X(K,I+1)-X(K,I)
2900 NEXTI
2910 :
2920 GOSUB3000
2930 :
2940 FORI=1TON-1
2950 B(I)=(A(I+1)-A(I))/H(I)-H(I)*(C(I+1)+2*C(I))/
3
2960 D(I)=(C(I+1)-C(I))/(3*H(I))
2970 NEXTI
2980 RETURN
2990 :
3000 REM KOEFFIZIENTENBERECHNUNG C
3010 REM MATRIX M
3020 :
3030 FORI=2TON-1
3040 M(I,I)=2*(H(I)+H(I+1))
3050 M(I,I+1)=H(I)
3060 M(I+1,I)=H(I)
3070 NEXTI
3080 :
3090 REM VEKTOR V
3100 :
3110 FORJ=2TON-1
3120 V(J)=3*(A(J+1)-A(J))/H(J)-3*(A(J)-A(J-1))/H(J
-1)
3130 NEXTJ
3140 :
3150 REM GAUSS
3160 :
3170 O(2)=M(2,2):P(2)=M(2,3)/O(2):Q(2)=V(2)/O(2)
3180 FORI=3TON
3190 O(I)=M(I,I)-M(I,I-1)*P(I-1)
3200 P(I)=M(I,I+1)/O(I)
3210 Q(I)=(V(I)-M(I,I-1)*Q(I-1))/O(I)
3220 NEXTI
3230 FORI=NTO2STEP-1
3240 C(I)=Q(I)-P(I)*C(I+1)
3250 NEXTI
3260 RETURN
3270 :
3280 REM ERGEBNISSE SPEICHERN
3290 :
3300 G=G+1
3310 Z(G)=INT(X*10000+.5)/10000:T(G)=INT(Y*10000+.
5)/10000
3320 IFE1=0THEN3350
3330 IFT(G)>MYTHENMY=T(G)
3340 IFT(G)<NYTHENNY=T(G)
3350 RETURN
3360 :
3370 :
3380 REM TAB COPY
3390 :
3400 PRINT"■":PRINT:PRINT:PRINT:PRINTSPC(9)"DRUCK
ER EINGESCHALTET ?"
3410 IFPEEK(203)<>60THEN3410
3420 OPEN4,4
3430 FORK=1TOAK:N=N1(K):G=0:E6=2:GOSUB2610
3440 PRINT#4,CHR$(18)"  KURVE "K"                X
     Y        "
3450 FORI=1TOG
3460 PRINT#4,CHR$(146)CHR$(16)"18"Z(I)  CHR$(16)"3
3"T(I)
3470 NEXTI
3480 IFE5<>0THEN3500
3490 NEXTK
3500 IFE5=1THEN5120
3510 IFE5=2THENE5=1:E6=0
3520 CLOSE4:GOTO280
3530 :
3540 REM EINGABE TAPE DISK
3550 :
3560 D=1:GOTO3580
3570 D=8
3580 PRINT"■"
3590 FORI=1TO6:PRINT:NEXTI
3600 IFD=1THENPRINTSPC(6)"RECORDER O.K. ?"
3610 IFD=8THENPRINTSPC(6)"FLOPPY O.K. ?"
3620 PRINT:PRINT:INPUT"          FILENAME";N$
3630 IFN$="←"THEN150
3640 PRINT:PRINT
3650 OPEN1,D,0, N$:OPEN2,8,15:INPUT#2,F,B$
3660 IFF=0THEN3700
3670 PRINT"■■  FEHLER:  "B$:CLOSE2:CLOSE1:DIR"$
3680 GETA$:IFA$<>""THENPRINT"■":GOTO3620
3690 GOTO3680
3700 INPUT#1,AK,SW,DI
3710 INPUT#1,T1$,MX,NX,V,TX,SX,E2
3720 INPUT#1,T2$,MY,NY,W,TY,SY,E3
3730 FORK=1TOAK
3740 INPUT#1,N1(K)
3750 IFN1(K)>N1THENN1=N1(K)
3760 NEXTK
3770 N1=N1+1
3780 DIMX(AK,N1),Y(AK,N1),SW(K,N1)
3790 DIMA(N1),B(N1),C(N1),D(N1),M(N1,N1),V(N1),O(N
1),P(N1),Q(N1)
3800 FORK=1TOAK
3810 FORI=1TON1(K)
3820 INPUT#1,X(K,I),Y(K,I):INPUT#1,SW(K,I)
3830 NEXTI
3840 NEXTK
3850 CLOSE1:CLOSE2:DIMZ(DI+3),T(DI+3):GOTO260
3860 :
3870 REM AUSGABE TAPE DISK
3880 :
3890 D=1:GOTO3910
3900 D=8
3910 PRINT"■"
3920 FORI=1TO6:PRINT:NEXTI
3930 IFD=1THENPRINTSPC(6)"RECORDER O.K. ?"
3940 IFD=8THENPRINTSPC(6)"FLOPPY O.K. ?"
3950 PRINT:PRINT:INPUT"          FILENAME";N$
3960 OPEN1,D,1,N$
3970 PRINT#1,AK:PRINT#1,SW:PRINT#1,DI:PRINT#1,T1$
3980 PRINT#1,MX:PRINT#1,NX:PRINT#1,V:PRINT#1,NX:PR
INT#1,SX:PRINT#1,E2
3990 PRINT#1,T2$:PRINT#1,MY:PRINT#1,NY:PRINT#1,W:P
RINT#1,NY
4000 PRINT#1,SY:PRINT#1,E3
4010 FORK=1TOAK:PRINT#1,N1(K):NEXTK
4020 FORK=1TOAK
4030 FORI=1TON1(K)
4040 PRINT#1,X(K,I):PRINT#1,Y(K,I):PRINT#1,SW(K,I)
4050 NEXTI:NEXTK
4060 CLOSE1:GOTO280
4070 :
4080 REM TEXT
4090 :
4100 PRINT"■■              SPLINE BERECHNUNG
   ■"
4110 FORI=1TO10:PRINT:NEXTI
4120 PRINTSPC(13)"BITTE WARTEN"
4130 RETURN
4140 :
4150 :
4160 REM BUBBLESORT
4170 :
4180 FORO=2TON1(I)
4190 FORP=N1(I)TOOSTEP-1
4200 IFX(I,P-1)>X(I,P)THEN4220
4210 GOTO4250
4220 H=X(I,P):Q=Y(I,P)
4230 X(I,P)=X(I,P-1):Y(I,P)=Y(I,P-1)
4240 X(I,P-1)=H:Y(I,P-1)=Q
4250 NEXTP
4260 NEXTO
4270 RETURN
4280 :
4290 :
4300 REM AUSGABE DATENFILE
4310 :
4320 PRINT:PRINT:PRINTSPC(14)"TAPE (1)"
4330 PRINT:PRINTSPC(14)"DISK (2)"
```

64ER ONLINE

Listing »Spline« (Fortsetzung)

```
4340 GETA$:A=VAL(A$):ONAGOTO 4350,4360:GOTO4340
4350 D=1:GOTO4370
4360 D=8
4370 PRINT""
4380 FORJ=1TO6:PRINT:NEXTJ
4390 IFD=1THENPRINTSPC(6)"RECORDER O.K. ?"
4400 IFD=8THENPRINTSPC(6)"FLOPPY O.K. ?"
4410 PRINT:PRINT:INPUT"         FILENAME ";N$
4420 OPEN1,D,1,N$
4430 PRINT#1,SW:PRINT#1,DI:PRINT#1,N1(I)
4440 FORK=1TON1(I)
4450 PRINT#1,X(I,K):PRINT#1,Y(I,K):PRINT#1,SW(I,K)
4460 NEXTK
4470 OPEN2,8,15:INPUT#2,F,B$
4480 IF F=0THEN4500
4490 PRINT"  FEHLER: "B$:CLOSE2:CLOSE1:GOTO4410
4500 CLOSE2:CLOSE1:D=0:GOTO890
4510 :
4520 :
4530 REM EINGABE DATENFILE
4540 :
4550 D=1:GOTO4570
4560 D=8
4570 PRINT""
4580 FORJ=1TO6:PRINT:NEXTJ
4590 IFD=1THENPRINTSPC(12)"RECORDER O.K. ?"
4600 IFD=8THENPRINTSPC(12)"FLOPPY O.K. ?"
4610 GETA$:IFA$=""THEN4610
4620 PRINT"":PRINT:INPUT"          ANZAHL DATENFILE
S";AK$
4630 IFAK$="←"THEN150
4640 AK=VAL(AK$)
4650 DIMN$(AK):PRINT
4660 FORJ=1TOAK
4670 IFJ<=0THENRUN
4680 PRINT:PRINT"          FILENAME";J:INPUT"
";N$(J)
4690 IFN$(J)="←"THENJ=J-2:NEXTJ
4700 NEXTJ
4710 FORJ=1TOAK
4720 OPEN1,D,0,N$(J):OPEN2,8,15:INPUT#2,F,B$
4730 IFF=0THEN4770
4740 PRINT"  FEHLER: "B$:CLOSE2:CLOSE1:DIR"$
4750 GETA$:IFA$<>""THENPRINT"":GOTO4660
4760 GOTO4750
4770 INPUT#1,SW,DE,N1(J)
4780 IFN1(J)>N1THENN1=N1(J)
4790 CLOSE1:CLOSE2:NEXTJ
4800 N1=N1+1
4810 DIMX(AK,N1),Y(AK,N1),SW(AK,N1)
4820 DIMA(N1),B(N1),C(N1),D(N1),H(N1),M(N1,N1),V(N
1),O(N1),P(N1),Q(N1)
4830 FORJ=1TOAK
4840 OPEN1,D,0,N$(J)
4850 INPUT#1,SW,DE,N1(J)
4860 IFDE>DITHENDI=DE
4870 FORK=1TON1(J)
4880 INPUT#1,X(J,K),Y(J,K),SW(J,K)
4890 NEXTK:CLOSE1
4900 NEXTJ:IFZW>DITHENDI=ZW
4910 DIMZ(DI+3),T(DI+3):D=0:GOTO260
4920 :
4930 REM VERGLEICH FKT - SPL
4940 :
4950 PRINT"":PRINT:PRINT:INPUT"      FUNKTION F(X
)=";F$
4960 IFF$="←"THENRUN
4970 PRINT"5140 DEFFNF(X)="F$
4980 PRINT"GOTO5000"
4990 POKE631,19:POKE632,13:POKE633,13:POKE198,3:EN
D
5000 PRINT"                FUNKTIONSDATEN
     "
5010 PRINT:PRINT:INPUT"  ANFANGS X-WERT";XA$
5020 IFXA$="←"THEN4950
5030 XA=VAL(XA$)
5040 PRINT:PRINT:INPUT"  END X-WERT";XE$
5050 IFXE$="←"THEN5010
5060 XE=VAL(XE$)
5070 PRINT:PRINT:INPUT"  ANZAHL ZWISCHENWERTE";ZW$
5080 IFZW$="←"THEN5040
5090 ZW=VAL(ZW$)
5100 SV=(XE-XA)/ZW:E5=1
5110 GOTO130
5120 G=0:E5=2
5130 FORX=XATOXE+.0001STEPSV
5140 DEFFNF(X)=X+2
5150 Y=FNF(X):GOSUB3300
5160 NEXTX
5170 IFE6=1THEN1050
5180 IFE6=2THEN3440
5190 GOTO2340
READY.
```

Listing »Spline« (Schluß)

Dem C 64 und Plus/4 Arbeit aufstapeln

Wollen Sie viele verschiedene Tätigkeiten direkt hintereinander ausführen und haben keine Lust, die ganze Zeit neben dem Computer sitzen zu müssen? Die Lösung heißt Batch- oder Stapel-Verarbeitung. Zusätzlich stellen wir Ihnen noch einen komfortablen Editor zur Verfügung.

Hier werden gleich zwei Programme auf einmal vorgestellt, die beide sehr nützlich sind:

EDI.BAS – ein total in Basic geschriebener und trotzdem vollwertiger, schneller Screen-Editor

BATCH – ein Programm, das einen Hauch von Großrechner-Komfort zugänglich macht, indem ganze Folgen von Programmen und Eingaben automatisch ablaufen, ohne daß der Benutzer Hand anlegen muß.

Die Batch-Verarbeitung arbeitet rein theoretisch wie ein sehr großer Tastaturpuffer. Man kann dem Computer im voraus die Eingaben für die nächsten Minuten oder gar Stunden geben. In diesen Eingaben können selbstverständlich LOAD- und RUN-Befehle enthalten sein, so daß sich ganze Programmketten auf einen Befehl hin abarbeiten lassen. Da man sich das wie einen Stapel von Eingaben vorstellen kann, von dem der Computer sich immer eine neue Arbeit abholt, bis der Stapel abgetragen ist, nennt man diese Prozedur Stapel- oder englisch Batch-Verarbeitung.

64ER ONLINE

Der Editor – EDI.BAS

Wie oft muß in Programmen eine größere Menge Text eingegeben werden oder es werden Datenfiles gebraucht, in denen bestimmte Werte drinstehen oder korrigiert werden sollen. Man findet dann oft mehr oder weniger aufwendig gestaltete Editor-Routinen, die aber lange nicht an den Komfort heranreichen, den ein echter Screen-Editor bietet.

Die Routine EDI.BAS, die im Listing 1 die Zeilen 1 bis 299 umfaßt, ist ein kleiner Auszug aus einem kompletten Textsystem. Sie ist völlig eigenständig und kann auch ohne den Rest dieses Listings verwendet werden.

Eine besondere Eigenschaft von EDI.BAS ist, daß es sich automatisch an 40- und 80-Zeichen-Schirme anpassen kann! Wen die Verfahrensweise interessiert, der sollte sich die Zeile 350 ansehen. Dort wird ein Zeichen an die 41. Bildschirmposition gesetzt und dann überprüft, ob es sich schon in der zweiten Zeile befindet.

Vor der ersten Benutzung als eigenständiges Programm ist EDI.BAS durch Aufruf der Zeile 200 zu initialisieren. Dies entfällt, wenn man das komplette Listing abgetippt hat.

EDI.BAS bearbeitet einen zeilenweise in A$() abgespeicherten Text. Aufgerufen wird der Editor nach der Initialisierung einfach mit GOSUB 100.

online.de online.net

EDI.BAS erklärt sich durch seine Help-Funktion fast von selbst. Diese Funktion kann man durch Eintippen von »H« und Return in der Kommandozeile erreichen.

Einschränkungen bestehen nur darin, daß nur die allernotwendigsten Funktionen vorhanden sind und daß die Zeilenlänge auf 77 Zeichen beschränkt werden mußte.

Erst dadurch, das EDI.BAS vollkommen in Basic programmiert wurde, ohne POKE- und PEEK-Befehle zu benutzen, konnte erreicht werden, daß die Routine auf fast allen Commodore-Rechnern, so dem C 64, C 128 oder dem 8032 anstandslos läuft.

Batch-Verarbeitung

Viele Aktionen sind nicht direkt programmierbar, so zum Beispiel die Aufrufe mehrerer Programme hintereinander und die Versorgung dieser Programme mit diversen Eingaben. Solche Befehlsfolgen können mit BATCH in einem Zug eingetippt und dann automatisch durchgeführt werden. Bedient wird BATCH folgendermaßen:

a) Start einfach mit RUN (ohne Parameter): Es wird ein Filename erfragt, unter dem die Kommandofolge auf der Floppy abgelegt werden soll beziehungsweise schon vorhanden ist. Wenn das File noch nicht existiert, kann es mit dem in BATCH 64 eingebauten Editor erstellt werden. Ist die Kommandofolge fertig, wird sie abgespeichert. Sie wird anschließend nochmals eingelesen und in einem geschützten RAM-Bereich abgelegt. Zum Schluß kann der Batch-Job mit einem unter dem Cursor bereitgestellten SYS-Befehl gestartet werden.

b) Start mit Angabe von Parametern: In einem Kommando-Mutterfile können manche Eingaben (zum Beispiel Filenamen) mit formalen Parametern %a, %b, ..., %z gekennzeichnet werden. Beim Aufruf von BATCH 64 werden diese dann mit aktuellen Inhalten versorgt und gelangen so zur Ausführung. Dazu muß hinter dem RUN-Befehl ein Doppelpunkt geschrieben werden, gefolgt von der Liste der aktuellen Parameter-Inhalte (Strings), jeweils durch Kommata getrennt. Der erste Parameter ist immer der Name des Kommando-Mutterfiles; wird hier ein Minus (-) eingegeben, wird der Parameter erfragt. Die folgenden entsprechen der alphabetischen Reihenfolge %a, ... Soll ein Parameterstring selbst ein Komma enthalten, muß er in Gänsefüßchen eingeschlossen werden. Gänsefüßchen können nicht übergeben werden! Beispiel: RUN: Mutterfile, erster Par, zweiter,, vierter, "ein komma: ,"

Bleibt ein Parameter leer (hier %c), erzeugt er im Kommandofile eine leere Stelle. Auf diese Weise wird aus dem Kommando-Mutterfile ein aktuelles Kommandofile (Batchfile) erzeugt, das auf Wunsch auch noch verändert werden kann. Vor Ausführung muß es auf jeden Fall noch einmal auf Floppy gespeichert werden. Der endgültige Aufruf erfolgt mit SYS wie oben bei a).

c) Aufruf mit Angabe des Kommandofilenamens: Ist schon ein fertig verwendbares Kommandofile vorhanden, kann es ohne weitere Nachfragen direkt gestartet werden. Dazu schreibt man dessen Filenamen (und nur diesen) durch Doppelpunkt getrennt hinter das RUN des Startkommados für das Programm BATCH 64. Beispiel: RUN:kommando.bat (also ähnlich wie oben das Mutterfile).

Wirkungsweise:

Der Kommandotext wird zusammen mit einer zugehörigen Maschinen-Routine unterhalb des aktuell zugänglichen RAM-Endes abgelegt (relokiert). Dieser Speicherbereich wird durch Umsetzen des Top-of-RAM-Pointers vor Überschreiben durch Basic geschützt. Die Maschinenroutine wird in die Interrupt-Routine des Computers eingeschleift. Stellt diese nach Aktivierung fest, daß der Tastaturpuffer leer ist,

```
0 REM SAVE"@0:BATCH64",8:REM BATCH-VERARBE
  ITUNG AUF C 64 , VERWENDET:               <100>
1 REM  EDI.BAS  ZEILENORIENTIERTER ASCII-T
  EXT-EDITOR FUER ALLE CBM-RECHNER          <135>
2 REM  VON P. KITTEL , FFM 1985             <075>
4 GOTO 200:               REM  COLD START MI
  T INITIALISIERUNG                         <063>
20 BJ=8:BI=CS+1:IF BI>LEN(A$)THEN BI=LEN(A
   $):IF BI=0 THEN BI=1                     <028>
22 GOSUB 28:CS=BI-1:IF B1<>13 AND BI<=ML A
   ND((B1<>17)OR K)THEN 22                  <114>
24 IF B1=13 THEN RETURN                     <106>
25 IF B1=17 AND K=0 THEN K=B:RETURN         <064>
26 IF LEN(A$)>250 THEN A$=LEFT$(A$,LEN(A$)
   -1):IF BI>LEN(A$)THEN BI=BI-1:PRINT"{LE
   FT}";                                    <176>
27 PRINT BL$;:GOTO 22                       <123>
28 B1=0:BJ$=MID$(A$,BI,1):BI$=MID$(A$,BI+1
   ,1):IF BJ$=""THEN BJ$=" "                <036>
29 IF(ASC(BJ$)AND 127)=18 THEN BJ$=CHR$(AS
   C("#")+(ASC(BJ$)AND 128))                <158>
30 GET B$:IF B$<>""THEN PRINT"{RVOFF}"BJ$;
   :GOTO 34                                 <248>
31 BJ=(1+BJ)AND 15:PRINT CHR$(18+16*(BJ AN
   D 8));BJ$;                               <196>
32 IF BJ$=G$THEN PRINT G$"{LEFT,RVOFF}";:I
   F BI<LEN(A$)AND BI$<>G$THEN PRINT BI$"{
   LEFT}";                                  <116>
33 PRINT"{LEFT}";:RETURN                    <046>
34 IF BJ$=G$THEN PRINT G$"{LEFT}";:IF BI<L
   EN(A$)AND BI$<>G$THEN PRINT BI$"{LEFT}"
   ;                                        <207>
35 PRINT"{LEFT}";:B=ASC(B$):B1=B AND 127:I
   F B1>31 THEN 39                          <163>
36 IF B1>12 THEN ON B1-12 GOTO 74,38,38,38
   ,38,87,76,60,38,38,38,38,38,38,38,38,46 <006>
37 REM 13:RETURN,17:VERT,18:REV,19:HOME,20
   :DEL,29:HORIZ                            <189>
38 RETURN                                   <096>
39 IF BI>ML THEN PRINT BL$;:RETURN          <109>
40 BI=BI+1:IF B=222 THEN B$=CHR$(255)       <014>
41 PRINT B$;:IF B$=G$THEN PRINT G$"{LEFT}"
   ;:IF BI<=LEN(A$)AND BI$<>G$THEN PRINT B
   I$"{LEFT}";                              <104>
42 IF BI>2 THEN B$=LEFT$(A$,BI-2)+B$        <146>
43 B$=B$+MID$(A$,BI)                        <086>
44 A$=B$:RETURN                             <121>
46 IF B=29 THEN 54                          <157>
48 IF BI>1 THEN PRINT B$;:BI=BI-1           <128>
50 IF BI<2 THEN PRINT BL$;                  <253>
52 RETURN                                   <110>
54 IF BI>=ML THEN PRINT BL$;:RETURN         <102>
56 PRINT B$;:BI=BI+1:IF LEN(A$)<BI THEN A$
   =A$+" "                                  <100>
58 RETURN                                   <116>
60 IF B=148 THEN 66                         <225>
62 IF BI<2 THEN PRINT BL$;:RETURN           <243>
63 PRINT B$;:B$=MID$(A$,BI):IF BI>2 THEN B
   $=LEFT$(A$,BI-2)+B$                      <037>
64 A$=B$:BI=BI-1:RETURN                     <175>
66 L=LEN(A$):BJ$=RIGHT$(A$,1):IF L>=ML THE
   N PRINT BL$;:RETURN                      <106>
68 PRINT B$"{SPACE,LEFT}";:B$=" "+MID$(A$,
   BI):IF BI>1 THEN B$=LEFT$(A$,BI-1)+B$    <087>
72 A$=B$:RETURN                             <149>
74 PRINT:IF B=13 THEN B=17:B1=B:BI=1:GOTO
   81                                       <068>
75 B=141:RETURN                             <121>
76 IF B=19 THEN PRINT LEFT$(CB$,BI-1);:BI=
   1:RETURN                                 <001>
78 IF BI<LEN(A$)THEN PRINT SPC(LEN(A$)-BI)
   ;:BI=LEN(A$)                             <121>
79 IF BI<ML AND RIGHT$(A$,1)<>" "THEN BI=B
   I+1:A$=A$+" ":PRINT"{RIGHT}";            <102>
80 RETURN                                   <138>
81 IF M>=N THEN RETURN                      <078>
82 B$=A$(M+1):IF B$=""THEN RETURN           <244>
83 IF MID$(B$,BI,1)=" "THEN IF BI<LEN(B$)T
   HEN BI=BI+1:GOTO 83                      <170>
84 IF MID$(B$,BI,1)=" "THEN BI=1            <071>
85 RETURN                                   <143>
87 BI=BI+1:IF BI>ML+1 THEN PRINT BL$;       <181>
88 PRINT CHR$(ASC("#")+(B AND 128));:GOTO
   42                                       <085>
90 PRINT"LX (X = ZAHL) = LOESCHEN{2SPACE}X
    ZEILEN OD.                              <003>
91 PRINT"EX (X = ZAHL) = EINFUEGEN X ZEILE
   N OD.                                    <200>
```

```
92 PRINT"H{2SPACE}FUER HILFE ODER":PRINT"-
   {2SPACE}FUER ABBRUCH.                     <121>
93 PRINT"{DOWN}IN DEN TEXTZEILEN:":PRINT"C
   URSORTASTEN UND INST/DEL NORMAL           <042>
94 PRINT"SHIFT-RETURN{2SPACE}= ZURUECK ZU
   BEFEHLSZEILE                              <246>
95 PRINT"RETURN{8SPACE}= NAECHSTER ZEILENA
   NFANG                                     <247>
96 PRINT"HOME{10SPACE}= AKTUELLER ZEILENAN
   FANG                                      <228>
97 PRINT"SHIFT-HOME{4SPACE}= ZEILENENDE.     <042>
99 PRINT"{DOWN}TASTE DRUECKEN.":FOR I=-1 T
   O 0:GET A$:I=A$="":NEXT                   <002>
100 K=0:PRINT"{CLR}":N=-N*(N>=0):G$=CHR$(3
    4):BL$=CHR$(7)                           <003>
101 PRINT"{HOME}";TAB(69);M-(M<0)+(M-N)*(M
    >N)                                      <016>
102 PRINT"{HOME}AEND/{RVSON}L{RVOFF}OESCH/
    {RVSON}E{RVOFF}INFUEG/{RVSON}H{RVOFF}I
    LFE, ENDE MIT - , ";                     <135>
103 INPUT"ZEILENNR./BEFEHL(L./E./H) = ";A$
    :B$=LEFT$(A$,1):IF B$="-"THEN RETURN     <207>
104 V=M:IF M<0 THEN V=0                      <141>
105 B=VAL(MID$(A$,2)):IF B$="L"THEN B=B+V-
    1:GOSUB 185:M=V:GOTO 100                 <141>
106 M=VAL(A$):IF B$="E"THEN 190              <151>
110 IF M<0 OR M>N THEN 100                   <150>
111 PRINT"{HOME}CURSOR AKTIV,{2SPACE}RETUR
    N: NAECHSTE ZEILE,{2SPACE}KOMMANDOZ.:"
    ;                                        <134>
112 PRINT" SHIFT-RETURN, Z.:"                <251>
113 ES$="."+CHR$(20)+LEFT$(CB$,50):FOR I=1
     TO 39-40*C8:PRINT"-";:NEXT:PRINT        <137>
114 IF B$="H"THEN PRINT"{DOWN}BEFEHLSZEILE
    :":PRINT"ZU AENDERNDE ZEILENNR. ODER":
    GOTO 90                                  <082>
118 I=4-6*C8:ZV=M-I:ZB=M+I+1:IF ZV<0 THEN
    ZV=0:ZB=2*I+1                            <100>
119 FOR J=ZV TO ZB:IF J>N THEN A$="":GOTO
    122                                      <235>
120 A$=A$(J)                                 <170>
122 L=LEN(A$):IF L=0 THEN PRINT LEFT$(CP$,
    78)ES$:GOTO 128                          <223>
124 FOR BI=1 TO L:B$=MID$(A$,BI,1):IF B$=G
    $THEN PRINT G$;CHR$(20);                 <143>
125 IF(ASC(B$)AND 127)=18 THEN B$=CHR$(ASC
    ("#")+(ASC(B$)AND 128))                  <030>
126 PRINT B$;:NEXT                           <110>
127 PRINT LEFT$(CP$,78-L);ES$                <006>
128 NEXT                                     <138>
130 PRINT"{HOME}":FOR I=ZV TO M:PRINT:NEXT   <017>
155 A$=A$(M):L=LEN(A$):IF CS>L-1 THEN CS=L
    -1:IF CS<0 THEN CS=0                     <052>
156 PRINT SPC(CS);                           <188>
160 GOSUB 20                                 <106>
161 IF LEN(A$)<2 THEN 169                    <252>
162 IF RIGHT$(A$,1)=" "THEN A$=LEFT$(A$,LE
    N(A$)-1):GOTO 161                        <234>
169 A$(M)=A$:IF K<=0 THEN 180                <194>
170 M=M+1+2*(K>20):IF M<ZV OR M>ZB THEN 10
    0                                        <248>
171 PRINT"{HOME}"TAB(69);M;"{LEFT,3SPACE}"
    :K=0:IF M>N THEN N=M                     <131>
172 GOTO 130                                 <148>
180 IF A$(N)=""THEN N=N-1:IF N>-1 THEN 180   <012>
182 GOTO 100                                 <110>
185 IF V<0 OR V>B OR B>N THEN PRINT"{DOWN}
    UNMOEGLICH!":FOR I=1 TO 200:NEXT:RETUR
    N                                        <123>
186 M=B-V+1:IF B=N THEN 188                  <009>
187 FOR I=B+1 TO N:A$(I-M)=A$(I):NEXT        <063>
188 N=N-M:IF N<MZ THEN FOR I=N+1 TO N+M:A$
    (I)="":NEXT:RETURN                       <143>
190 IF B+N>MZ OR B<1 THEN V=-1:GOSUB 185:G
    OTO 100                                  <234>
191 FOR I=N TO V STEP-1:A$(I+B)=A$(I):NEXT   <134>
192 FOR I=V TO V+B-1:A$(I)="":NEXT:IF CS=0
     OR CS>=LEN(A$(V+B))THEN 199             <159>
193 A$(V)=LEFT$(A$(V+B),CS):A$(V+B)=MID$(A
    $(V+B),CS+1)                             <167>
199 N=N+B:M=V:GOTO 100                       <057>
200 REM INITIALISIERUNG EDI.BAS: ----------
    ------------------------------------     <086>
210 REM EINFUEHRUNG DER BENUTZTEN VARIABLE
    N:                                       <248>
220 DIM A$,B,B$,B1,BI,BJ,BJ$,G$,BI$,BL$,CB
    $,CP$,CS,ES$,I,J,K,L,M,ML,MZ,N,V,C8      <037>
221 DIM ZB,ZV                                <051>
230 MZ=600: REM  MAXIMALE ZEILENZAHL         <152>
240 ML= 77: REM  MAX. ZEICHENZAHL/ZEILE  G
    GF. PER INPUT (NUR KLEINER MOEGLICH)     <021>
250 N = -1: REM  AKTUELLER FUELLSTAND IN A
    $()                                      <147>
251 M = -1: REM  ZULETZT EDIERTE ZEILENNR.   <099>
262 CB$="{4LEFT}":FOR I=1 TO 5:CB$=CB$+CB$
    :NEXT:CB$=CB$+LEFT$(CB$,127):REM CURS
    BACK                                     <195>
263 CP$="{4SPACE}":FOR I=1 TO 5:CP$=CP$+CP
    $:NEXT:CP$=CP$+LEFT$(CP$,127):REM SPAC
    E                                        <152>
270 DIM A$(MZ) :REM  TEXT ZEILENWEISE        <235>
280 :           REM  EDITOR MIT GOSUB100 AU
    FRUFEN                                   <213>
284 :           REM  TEST AUF 40- BZW. 80-Z
    EICHEN-SCHIRM:                           <026>
285 REM PRINT"{2HOME,CLR,CTRL-N}"SPC(40)"T
    {HOME,DOWN}";:OPEN1,3:GET#1,A$:CLOSE1:
    C8=A$=" ":REM S.U.                       <162>
290 REM                                      <098>
299 REM EDI.BAS FERTIG --------------------
    ------------------------------------     <202>
300 PRINT"{UP,2RIGHT}";:DIM P$(27),N%(27):
    NP=0:OPEN 1,3:REM PARAMETER HOLEN        <164>
301 C=0:FOR I=2 TO 79:GET#1,A$:IF A$=":"TH
    EN C=I:I=80                              <243>
302 NEXT:IF C=0 THEN CLOSE 1:MP=-1:GOTO 35
    0                                        <136>
310 MP=0:FOR P=1 TO 27:G$="":FG=1:FC=0       <006>
311 FOR I=-1 TO 0:I=-1:GET#1,A$:IF A$=CHR$
    (34)THEN FG=1-FG:GOTO 320                <059>
312 C=C+1:IF C>79 OR A$=CHR$(13)THEN I=0:P
    $(FC*P)=G$:MP=MP+FC*(P-MP):P=99:GOTO 3
    20                                       <157>
313 IF FG THEN IF A$=","THEN I=0:P$(FC*P)=
    G$:MP=MP+FC*(P-MP):GOTO 320              <137>
315 IF FC OR(A$<>" ")THEN FC=1:G$=G$+A$      <105>
320 NEXT:NEXT:CLOSE 1:IF MP=0 THEN 350       <156>
330 FOR P=1 TO MP:IF P$(P)=""THEN 335        <248>
331 P$=P$(P):L=LEN(P$)                       <056>
332 IF RIGHT$(P$,1)=" "THEN L=L-1:P$=LEFT$
    (P$,L):GOTO 332                          <123>
333 P$(P)=P$                                 <120>
335 P$(P-1)=P$(P):NEXT:P$(MP)="":MP=MP-1     <185>
340 NP=-1:FOR P=0 TO MP:IF P$(P)<>""THEN N
    P=P                                      <121>
341 NEXT:MP=NP:F$=P$(0):IF F$=""THEN F$="-
    "                                        <020>
349 REM ZEILE 285 HIERHER VERLEGT:           <035>
350 PRINT"{2HOME,CLR,CTRL-N}"SPC(40)"T{HOM
    E,DOWN}";:OPEN 1,3:GET#1,A$:CLOSE 1:C8
    =A$=" ":REM 40/80 ?                      <067>
351 PRINT"{CLR,DOWN,SPACE,RVSON,CTRL-N,SPA
    CE}B{SHIFT-SPACE}A{SHIFT-SPACE}T{SHIFT
    -SPACE}C{SHIFT-SPACE}H{SHIFT-SPACE,RVO
    FF,DOWN}                                 <073>
352 FP=0:IF MP=0 AND F$<>"-"AND F$<>""THEN
     FP=1:GOTO 511                           <228>
353 IF MP<0 THEN 500                         <170>
360 PRINT"{2DOWN}PARAMETER:":PRINT"{DOWN}F
    ILE: "F$"{DOWN}"                         <117>
361 FOR P=1 TO MP:IF P$(P)<>""THEN PRINT"{
    3SPACE}"CHR$(P+64)": "P$(P)              <018>
362 NEXT:GOTO 510                            <233>
500 PRINT"{DOWN}NAMEN FUER SEQ-KOMMANDO-FI
    LE EINGEBEN,                             <202>
502 PRINT "WENN NOCH KEINES EXISTIERT: EIN
    FACH{2SPACE}-                            <094>
505 INPUT"{DOWN}FILENAME = ";F$              <129>
510 IF F$="-"THEN 520                        <239>
511 PRINT"{DOWN}FILE WIRD EINGELESEN, BITT
    E GEDULD.{3DOWN}                         <006>
512 N$=CHR$(0):CR$=CHR$(13):N=-1:OPEN 1,8,
    3,F$                                     <090>
513 FOR Z=-1 TO 0:G$="":FOR I=-1 TO 0:GET#
    1,A$:S=ST:A$=LEFT$(A$+N$,1)              <219>
514 I=(S=0)AND(A$<>CR$):IF A$<>CR$THEN G$=
    G$+A$:IF A$="%"THEN IF MP>0 THEN GOSUB
     1000                                    <197>
515 NEXT:N=N+1:A$(N)=G$:PRINT G$:Z=S=0:NEX
    T:CLOSE 1:IF MP>0 THEN 520               <215>
516 IF FP THEN 650                           <184>
517 PRINT"{3DOWN}AENDERUNGEN NOETIG (J/N)
    ? ";                                     <231>
```

Listing 1. Programm EDI.BAS/BATCH. Bitte mit dem Checksummer Seite 6 eingeben.

```
518 GET A$:IF A$<>"J"AND A$<>"N"THEN 518      <040>
519 PRINT A$:IF A$="N"THEN 650                <118>
520 PRINT"{DOWN}FILE MUSS MIT EINGEBAUTEM
    EDITOR                                    <168>
521 PRINT "BE/ER-ARBEITET WERDEN.             <015>
522 PRINT "GEBEN SIE IN DER SPAETEREN KOMM
    ANDO-                                     <222>
523 PRINT "ZEILE{SPACE,RVSON}H{RVOFF,SPACE
    }FUER HILFE EIN.                          <081>
524 PRINT"{DOWN}HINWEIS: MIT DIESEM EDITOR
     KOENNEN NUR                              <174>
525 PRINT "{9SPACE}STANDARDZEICHEN, CR UND
     RVS                                      <151>
526 PRINT "{9SPACE}EINGEGEBEN WERDEN,{SPAC
    E,RVSON}KEINE{RVOFF}                      <049>
527 PRINT "{9SPACE}ANDEREN STEUERZEICHEN!  <236>
528 PRINT"{DOWN}ALTERNATIVE: FILE MIT ANDE
    REM REINEN                                <040>
529 PRINT "{9SPACE}ASCII-EDITOR ERZEUGEN.  <203>
530 PRINT"{2DOWN}DRIVE FUER FILE-SPEICHERU
    NG (0/1) ? ";                             <002>
531 GET D$:IF D$<>"0"AND D$<>"1"THEN 531      <211>
532 PRINT D$:IF MP>0 THEN MP=0:GOTO 550       <208>
540 GOSUB 100                                 <232>
550 FA$=F$:PRINT"{2HOME,CLR,DOWN}FILE-INHA
    LT OK (J/N) ? ";                          <037>
551 GET A$:IF A$<>"J"AND A$<>"N"THEN 551      <063>
552 PRINT A$:IF A$="N"THEN 540                <215>
560 INPUT"{DOWN}FILE-NAME FUER SPEICHERUNG
     = ";F$:IF F$="-"THEN 560                 <010>
570 IF NP>0 THEN IF F$=FA$THEN PRINT"{DOWN
    }NICHT PARAMETER-FILE UEBERSCHREIBEN!"
    :GOTO 560                                 <054>
600 PRINT"{DOWN}FILE WIRD GESPEICHERT, BIT
    TE GEDULD.                                <033>
610 OPEN 1,8,15,"S"+D$+":"+F$:CLOSE 1:OPEN
     1,8,3,(D$+":"+F$+",S,W")                 <202>
620 FOR I=0 TO N:PRINT#1,A$(I):NEXT:CLOSE
    1                                         <191>
630 IF LEN(F$)<5 OR RIGHT$(F$,4)<>".MOT"TH
    EN 650                                    <069>
640 PRINT"{DOWN}BITTE MIT PARAMETERANGABE
    NEU STARTEN.":END                         <247>
650 M=0:FOR I=0 TO N:M=M+LEN(A$(I))+1:NEXT <157>
700 TR=55:REM PLUS/4: 55, 8XXX/40XX: 52       <005>
701 T=PEEK(TR)+256*PEEK(TR+1)-1:REM PLUS/4
    :  IFT>32767THENT=32767                   <211>
702 TH=INT(T/256):TL=T-256*TH                 <099>
703 LH=INT(M/256):LL=M-256*LH                 <172>
704 B=T-M:BH=INT(B/256):BL=B-256*BH           <107>
710 PA=BH:IF BL<122 THEN PA=PA-1              <063>
720 P=256*PA:POKE P,TL:POKE P+1,TH:POKE P+
    2,BL:POKE P+3,BH:POKE P+4,FP              <169>
721 L=LEN(F$):IF L>16 THEN L=16               <115>
722 FOR I=1 TO L:N%(I)=ASC(MID$(F$,I,1)):N
    EXT                                       <023>
730 POKE P+5,L:FOR I=1 TO L:POKE P+5+I,N%(
    I):NEXT                                   <114>
740 POKE 55,0:POKE 56,PA                      <087>
750 CLR                                       <100>
800 A$="":I=0:N$=CHR$(0)                      <205>
805 TR=55:REM PLUS/4: 55, 8XXX/40XX: 52       <112>
810 M=PEEK(TR+1):P=256*M                      <047>
815 TL=PEEK(P):TH=PEEK(P+1):BL=PEEK(P+2):B
    H=PEEK(P+3):FP=PEEK(P+4)                  <092>
820 L=PEEK(P+5):F$="":FOR I=1 TO L:F$=F$+C
    HR$(PEEK(P+5+I)):NEXT                     <061>
830 T=TL+256*TH                               <052>
835 PRINT"{DOWN}FILE "F$" WIRD NOCH EINMAL <071>
836 PRINT"EINGELESEN, BITTE GEDULD.{DOWN}    <168>
840 OPEN 1,8,3,F$:FOR I=-1 TO 0:GET#1,A$:I
    =ST=0:POKE T,ASC(A$+N$):T=T-1:NEXT:CLO
    SE 1                                      <183>
850 RESTORE:READ N                            <156>
860 FOR I=0 TO N:READ A:IF A<0 THEN A=M       <147>
861 POKE P+I,A:NEXT                           <000>
870 POKE P+ 71,TL:POKE P+ 72,TH               <003>
871 POKE P+118,BL:POKE P+119,BH               <229>
900 PRINT"{2DOWN}BATCH-JOB MIT{4SPACE}SYS"
    P"{4SPACE}STARTEN:                        <136>
910 IF FP THEN PRINT:SYS P:END                <154>
920 PRINT"{3DOWN}SYS"P"{3UP}"                 <224>
999 END                                       <239>
1000 GET#1,A$:S=ST:A$=LEFT$(A$+N$,1):A=ASC
     (A$)                                     <065>
1010 FOR P=1 TO MP:IF A=P+64 THEN G$=LEFT$
     (G$,LEN(G$)-1)+P$(P):P=MP:A=-1           <217>
1011 NEXT:IF A>-1 THEN G$=G$+A$               <173>
1020 I=(S=0)AND(A$<>CR$):RETURN               <090>
6000 A=7*4096:FOR Z=0 TO 13:PRINT A;"DATA"
     ;                                        <121>
6010 FOR I=0 TO 7:PE=PEEK(A):A=A+1:IF I>0
     THEN PRINT",";                           <022>
6020 PRINT" ";RIGHT$("00"+MID$(STR$(PE),2)
     ,3);:NEXT:PRINT:NEXT                     <062>
6030 END                                      <190>
7000 RESTORE:READ N:A=28672                   <096>
7010 FOR I=0 TO N:READ P:IF ABS(P)=112 THE
     N PRINT CHR$(18-(P<0)*128);A+(I AND 2
     48),                                     <022>
7020 NEXT:END                                 <133>
8000 REM"{2SPACE}BENUTZTE MASCHINENADRESSE
     N:                                       <234>
8010 REM"{2SPACE}55-56{4SPACE}POINTER TOP
     OF RAM                                   <060>
8020 REM"{2SPACE}$0314{4SPACE}POINTER IRQ-
     ROUTINE                                  <126>
8030 REM"{2SPACE}$0277{4SPACE}ERSTES BYTE
     TASTATURPUFFER                           <123>
8040 REM"{2SPACE}$C6{6SPACE}ANZAHL BYTES I
     M TASTATURPUFFER                         <250>
28600 REM  SOURCE-FILE:  IRQBAT.SRC           <050>
28666 DATA 119                                <169>
28672 DATA 120, 173, 020, 003, 141, 116,-1
      12, 173                                 <052>
28680 DATA 021, 003, 141, 117,-112, 169, 0
      51, 141                                 <155>
28688 DATA 020, 003, 169,-112, 141, 021, 0
      03, 088                                 <160>
28696 DATA 096, 189, 005, 001, 041, 252, 2
      01, 044                                 <060>
28704 DATA 208, 017, 189, 006, 001, 201, 2
      49, 208                                 <173>
28712 DATA 010, 169, 052, 157, 005, 001, 1
      69, 249                                 <039>
28720 DATA 157, 006, 001, 165, 198, 240, 0
      13, 201                                 <039>
28728 DATA 002, 176, 044, 173, 119, 002, 2
      01, 003                                 <189>
28736 DATA 240, 037, 208, 047, 230, 198, 1
      73, 255                                 <119>
28744 DATA 255, 141, 119, 002, 173, 071,-1
      12, 208                                 <087>
28752 DATA 003, 206, 072,-112, 206, 071,-1
      12, 173                                 <030>
28760 DATA 071,-112, 205, 118,-112, 208, 0
      20, 173                                 <250>
28768 DATA 072,-112, 205, 119,-112, 208, 0
      12, 173                                 <231>
28776 DATA 116,-112, 141, 020, 003, 173, 1
      17,-112                                 <124>
28784 DATA 141, 021, 003, 076, 255, 255, 0
      00, 000                                 <081>
```

Listing 1. Programm EDI.BAS/BATCH (Schluß)

holt sie aus dem Kommandotext das nächste Byte und legt es in den Tastaturpuffer. Die eigene Interrupt-Routine schaltet sich automatisch selbst ab, wenn:
- der ganze Kommandotext abgearbeitet ist,
- die Stopptaste gedrückt wird oder
- im Tastaturpuffer mehr als ein Tastendruck, also Eingaben des Benutzers, angetroffen werden.

Der Top-of-RAM-Pointer wird nicht zurückgesetzt!

Einschränkungen:

Es sind einerseits die RAM-Belegung und andererseits das Interrupt-Verhalten aufgerufener Software zu beachten. Durch geeignetes Setzen des Top-of-RAM-Pointers vor Aufruf von BATCH 64 kann meist eine Kollision mit anderen Programmen in diesem Speicherbereich vermieden werden.

Da eine eigene Interrupt-Routine eingeschleift wird, kann es bei Programmen, die das auch tun, zu Kollisionen kommen. Bei wiederholten Aufruf von BATCH64 wird der Top-of-RAM-Pointer immer nur noch tiefer gesetzt, so daß er in vernünftigen Abständen wieder initialisiert werden sollte, zum Beispiel durch POKE56,160:clr

Als Detail sei noch auf die Parameterübergabe im RUN-Befehl hingewiesen, die durch die GET-Funktion aus dem Bildschirm bewerkstelligt wird. Dadurch sind Schreibweisen möglich, wie man sie sonst nur aus Betriebssystemkomman-

```
                    line

00002  0000                ;put"B0:irqbat.src"
00003  0000                ;*
00004  0000                ;*****************************************************
00005  0000                ;*
00006  0000                ;* irq - bat
00007  0000                ;*
00008  0000                ;* batch-job-abwicklung ueber irq-routine
00009  0000                ;* version fuer 8xxx/40xx mit hinweisen fuer c 64, plus/4
00010  0000                ;*
00011  0000                ;* p. kittel, frankfurt 1985
00012  0000                ;*
00013  0000                ;*****************************************************
00014  0000                ;
00015  0000                irqptr = $fe90        ;pointer irq-routine (8xxx: $0090)
00016  0000                           ;                  high-byte $fe=254 in datas korrigieren!
00017  0000                           ;                  c-64: $0314,  plus/4: $0212
00018  0000                                 ;
00019  0000                tasanz = $9e          ;anzahl im tastaturpuffer (8xxx)
00020  0000                           ;                  c-64: $c6,    plus/4: $ef
00021  0000                           ;
00022  0000                taspuf = $026f        ;tastaturpuffer (8xxx)
00023  0000                           ;                  c-64: $0277,  plus/4: $0527
00024  0000                           ;
00025  0000                ;
00026  0000                      * = $7000
00027  7000                ;
00028  7000  78            aktiv  sei            ; batch-routine aktivieren
00029  7001  ad 90 fe             lda irqptr     ;alten pointer retten
00030  7004  8d 74 70             sta irqend+1
00031  7007  ad 91 fe             lda irqptr+1
00032  700a  8d 75 70             sta irqend+2
00033  700d                                      ;statt 'start' bei c-64 und plus/4    start1 :
00034  700d  a9 19                lda #<start    ;neuen setzen, bzw. start1
00035  700f  8d 90 fe             sta irqptr
00036  7012  a9 70                lda #>start    ;bzw. start1 (s.o.)
00037  7014  8d 91 fe             sta irqptr+1
00038  7017  58                   cli
00039  7018  60                   rts            ;fertig, gleich gehts los
00040  7019                ;
00041  7019                ;programmteil nur fuer cbm 8/4xxx benoetigt:
00042  7019  bd 05 01      start  lda $0105,x    ;i/o-ende-schleife?
00043  701c  29 fc                and #$fc       ;ruecksprungadr pruefen
00044  701e  c9 2c                cmp #$2c
00045  7020  d0 11                bne start1     ;ne: nein
00046  7022  bd 06 01             lda $0106,x
00047  7025  c9 f9                cmp #$f9
00048  7027  d0 0a                bne start1     ;ne: nein
00049  7029  a9 34                lda #$34       ;ruecksprungadr
00050  702b  9d 05 01             sta $0105,x    ;manipulieren
00051  702e  a9 f9                lda #$f9
00052  7030  9d 06 01             sta $0106,x
00053  7033                ;
00054  7033  a5 9e         start1 lda tasanz      ;tastenpuffer leer?
00055  7035  f0 0d                beq holtas     ;eq: ja, was dagegen tun
00056  7037  c9 02                cmp #2         ;mehr als eine taste?
00057  7039  b0 2c                bcs reset      ;cs: ja, abbruch wegen operator-eingriff
00058  703b                ;
00059  703b  ad 6f 02             lda taspuf     ;zuletzt gedrueckte taste
00060  703e  c9 03                cmp #3         ;stop-taste?
00061  7040  f0 25                beq reset      ;eq: ja, abbrechen
00062  7042  d0 2f                bne irqend     ;ne: einfach weiter


irqbat......page 0003

line# loc   code        line


00063  7044                ;
00064  7044  e6 9e         holtas inc tasanz      ;eine taste 'druecken'
00065  7046  ad ff ff      holptr lda $ffff       ;dummy-adresse wird modifiz.
00066  7049  8d 6f 02             sta taspuf     ;in tastaturpuffer
00067  704c  ad 47 70             lda holptr+1   ;low byte
00068  704f  d0 03                bne nocar      ;ne: kein uebertrag
00069  7051  ce 48 70             dec holptr+2   ;high byte
00070  7054  ce 47 70      nocar  dec holptr+1    ;naechstes byte
00071  7057                ;
00072  7057  ad 47 70             lda holptr+1   ;abfrage auf ende
00073  705a  cd 76 70             cmp endptr
00074  705d  d0 14                bne irqend     ;ne: nein
00075  705f  ad 48 70             lda holptr+2
00076  7062  cd 77 70             cmp endptr+1
00077  7065  d0 0c                bne irqend     ;ne: nein, weiter
00078  7067                ;
00079  7067  ad 74 70      reset  lda irqend+1    ;fertig, irq restaurieren
00080  706a  8d 90 fe             sta irqptr
00081  706d  ad 75 70             lda irqend+2
00082  7070  8d 91 fe             sta irqptr+1
00083  7073                                ;
00084  7073  4c ff ff      irqend jmp $ffff       ;dummy-adresse wird modifiziert
00085  7076                ;
00086  7076  00            endptr .byte 0,0       ;hier pointer auf (letztes byte -1)
00086  7077  00
00087  7078                ;
00088  7078                      .end
```

Listing 2. Der Sourcecode für BATCH 64

dos anderer, größerer Computer kennt.

Zur Anpassung an andere Computer sind die DATA-Zeilen mit der Maschinenroutine (siehe unten) und die Zeilen 700 und 805 mit der Adresse des Top-of-Memory-Pointers zu verändern. Letztere ist auch zu beachten, wenn man diesen Pointer ab und zu wieder restauriert:

```
POKE56,160:CLR:REM C 64
POKE56,128:CLR:REM PLUS/4
POKE53,128:CLR:REM 8xxx/40xx
```

Daß man beim PLUS/4 nur die ersten 32 KByte des Speichers ausnutzen kann (siehe auch Zeile 701, Listing 1!), liegt daran, daß man bei höherer Plazierung der eigenen Interrupt-Routine noch Bank-Switching zwischen parallelliegendem RAM und ROM ausführen müßte. Diesen Ausbau der Routine will ich gern interessierten Lesern überlassen.

Noch etwas zum Verfahren: Der Kommandotext muß ins RAM gelegt werden, weil zum Zeitpunkt der Abarbeitung durchaus schon alle Floppy-Kanäle (beziehungsweise Puffer) belegt sein können, so daß kein direkter Zugriff auf das Kommandofile mehr möglich ist. Der Top-of-RAM-Pointer darf auch nicht automatisch beim Ausklinken der Interrrupt-Routine zurückgestellt werden, denn anschließend müßte ja ein CLR-Befehl gegeben werden, der Variablen eines womöglich nicht in vollem Lauf befindlichen Basic-Programms einfach zerstören würde!

Theoretischerweise kann man in BATSCH 64 auch die Routine EDI-BAS weglassen und durch eine andere ersetzen, oder man benutzt einen externen Texteditor zur Erstellung der Kommandofiles. Auf jeden Fall muß das File anschließend reinen ASCII-Text enthalten, also nicht Bildschirmcode oder Vorspänne mit Tab- und Formatier-Informationen.

Listing 2 ist das Assembler-Listing der von BATCH verwendeten Maschinen-Routine, und zwar in der 8xxx/40xx-Version. Die Version für den C 64 können Sie den Hinweisen und den DATAs aus Listing 1 entnehmen. Hinweise gibt es auch zur Anpassung an den Plus/4.

Der Quelltext ist mit Gewalt (Adresse $fe90 statt $90 für 8xxx, das Byte $fe=254 muß nachträglich zu Fuß in den DATAs auf 000 korrigiert werden!) so gestaltet, daß er für alle Computer die gleiche Länge hat. Die Anpassung an andere kann also direkt in den DATA-Zeilen von Listing 2 erfolgen, wenn man im Assembler-Listing die Änderungen und Vorkommen der fraglichen Werte heraussucht.

Nur für 8xxx/40xx ist der Programmteil aktiv, der sich um die I/O-Ende-Routine kümmert. Dahinter verbirgt sich eine ganz böse Falle für jeden Interrupt-Routinen-Programmierer. Auf besagtem Rechner wird nämlich nach jeder Ein-/Ausgabe eine Routine aufgerufen, die einfach so lange wartet, bis der Interrupt-Pointer wieder seinen Normalwert hat! Das kommt von den Kassettenroutinen, deren Timing über eigene Interrupt-Routinen gemanagt wird. Wenn man den Pointer nun aber selbst verbogen hat und zum Beispiel einen LOAD-Befehl gibt, bleibt der Computer erst mal hängen, läßt sich aber immerhin mit der Stopptaste zurückholen. Zur Abhilfe schaut nun in unserer Routine ein Programmteil auf dem Stack nach, ob die Rücksprungadresse womöglich irgendwo in diese Warteschleife zeigt. Wenn ja, wird die Rücksprungadresse brutal auf den nächsten RTS-Befehl umgesetzt.

Testlauf

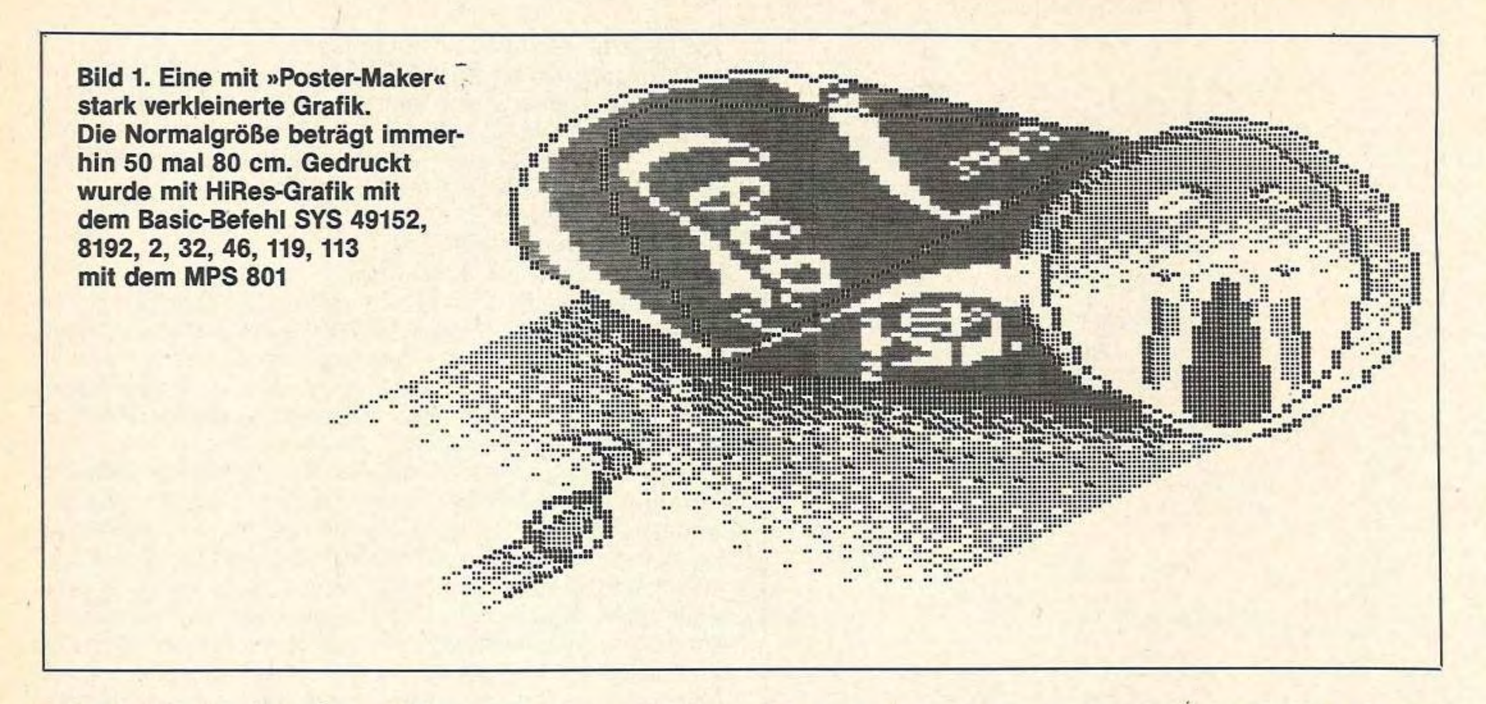
Erstellen Sie am besten ein Kommandofile namens TEST.BAT mit folgenden zwei Zeilen Inhalt (Kleinschreibung wegen Groß/Klein-Modus des Editors):

```
list-5
run:test-bat
```

Wenn Sie BATCH 64 mit RUN:TEST.BAT aufrufen, müßte sich ein immer wieder selbst anstartender Ablauf ergeben, der erst durch OUT OF MEMORY ERROR gebremst wird, wenn Top-of-Memory nach wenigen Stunden zu klein geworden ist.

(Dr. P. Kittels/bs)

64ER ONLINE


Bild 1. Eine mit »Poster-Maker« stark verkleinerte Grafik. Die Normalgröße beträgt immerhin 50 mal 80 cm. Gedruckt wurde mit HiRes-Grafik mit dem Basic-Befehl SYS 49152, 8192, 2, 32, 46, 119, 113 mit dem MPS 801

Poster-Maker für den C 64

Ein- und mehrfarbige HiRes-Grafiken können in einer Größe von sage und schreibe 50 x 80 cm auf fast jedem Drucker ausgegeben werden.

Dazu wird der Grafikbildschirm in vier gleich große, senkrechte Streifen zu je 80 x 200 Punkten unterteilt. Diese werden hintereinander auf Endlospapier ausgedruckt, wobei ein Grafikpunkt einem Druckzeichen entspricht. Bei Mehrfarbgrafiken besteht die Möglichkeit, für jedes der vier Bitmuster 00, 01, 10, 11, denen im Multicolor-Modus verschiedene Farben zugeordnet sind, verschiedene Druckzeichen zu definieren. Durch die Wahl dieser Zeichen lassen sich auch Graustufen erzeugen. Beim Ausdruck sind dann jeweils zwei nebeneinanderliegende Zeichen gleich.

Der Aufruf des Programms erfolgt durch einen SYS-Befehl:

SYS 49152,adr,m,c1,c2,c3,c4

Dabei ist 49152 die Startadresse des Programms ($C000), adr ist die Anfangsadresse der Grafik.

m ist der Grafikmodus. Soll eine Grafik im normalen Einzelpunktmodus (Auflösung 320 x 200 Punkte) ausgedruckt werden, ist m = 1 zu setzen. Bei einer Grafik im Multicolor-Modus (Auflösung 160 x 200 Punkte) ist m = 2. c1 bis c4 entsprechen den ASCII-Codes der verschiedenen Druckzeichen:

c1 entspricht Bitmuster 00
c2 entspricht Bitmuster 01
c3 entspricht Bitmuster 10
c4 entspricht Bitmuster 11

Dies gilt jedoch nur für den Multicolor-Modus. Im Einzelpunktmodus müssen c3 und c4 entfallen, da dort nur die beiden Bitmuster 0 und 1 exisitieren (c1 steht dann für 0, c2 für 1). Beispiel für einen Aufruf:

SYS 49152,8192,2,32,46,119,113

Damit erfolgt ein Ausdruck einer Mehrfarbgrafik (Bild 1 erstellt mit einem Commodore-Drucker beziehungsweise eines Ausschnitts Bild 2 erstellt mit einem FX-80), die bei 8192 beginnt, wobei 00 zwei Leerstellen, 01 zwei Punkte, 10 zwei nicht ausgefüllte Kreise und 11 zwei ausgefüllte Kreise zugeordnet sind.

Der Ausdruck einer Einzelpunktgrafik kann zum Beispiel so aussehen:

SYS 49152,8192,1,32,113

Es ist zu beachten, daß höchstens vier verschiedene Druckzeichen definiert werden können, auch wenn die Grafik mehr als vier Farben hat. Diese sind dann durch das Farbregister festgelegt. Das Programm unterscheidet jedoch nur zwischen den Bitmustern und nicht zwischen den Werten im Farbregister, die dem jeweiligen Bildpunkt entsprechen.

Das Programm ist so ausgelegt, daß es auf den meisten Druckern läuft. Das einzige, was an den Drucker angepaßt werden muß, sind die Kommandos zur Verkleinerung des Zeilenabstandes, um einen lückenlosen Ausdruck zu erreichen. Das Programm ist für folgende Drucker vorbereitet:

MPS 801/VC 1525
MPS 802/VC 1526
Epson und Kompatible

Bei allen anderen Druckern muß vor dem Aufruf des Programms der Zeilenabstand laut Druckerhandbuch selbst eingestellt werden.

Zur Handhabung des Basic-Laders

Das im Lader (siehe Listing) enthaltene Menü erlaubt es, verschiedene Parameter des Programms festzulegen und es dadurch an einen Drucker anzupassen.

Zunächst wird nach einer Filenummer (fn) und einer Sekundäradresse (sa) gefragt. Das Maschinenprogramm führt später folgenden Befehl aus: OPEN fn,4,sa

Hierbei kann man die Standardwerte einsetzen, das heißt die Werte, die man zum Beispiel beim Ausdruck eines Listings benutzt. Als drittes muß der Druckertyp eingegeben werden. Die jeweiligen Nummern werden im Menü aufgeführt.



Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Bestellungen aus der Schweiz bitte direkt an: Markt&Technik Vertriebs AG Kollerstrasse 3 CH-6300 Zug Tel. 042/41 56 56

Bestellungen aus Österreich bitte direkt an: Ueberreuter Media Handels- und Verlagsges. mbH, Alser Straße 24, 1091 Wien Tel. 0222/48 15 38-0

Hardware für alle - ein neuer 64'er Leser-Service

Der Commodore 64 hat schon oft bewiesen, wie vielseitig er ist. Er läßt sich nicht nur mit Programmen, sondern auch durch so manche Hardware-Erweiterung sinnvoll nutzen und ausbauen. Dabei ist es sicherlich ein reizvoller Bestandteil des Computer-Hobbys, sich solche Erweiterungen selbst nachzubauen. Aber nicht jeder Leser verfügt über die Gelegenheit und Zeit zur Platinenherstellung. Hinzu kommt, daß es oft zu teuer ist, wegen einer bestimmten Erweiterung, Investitionen von mehreren hundert Mark für eine Platinenstation zu tätigen. Wir haben reagiert: Ab sofort besteht die Möglichkeit, im Rahmen des Leser-Service, die in der 64'er abgedruckten Hardware-Erweiterungen in drei verschiedenen Ausbaustufen zu erhalten:

1. Als Platinen
Nur Leerplatinen. Die Beschaffung der Bauteile und der Zusammenbau bleibt bei Ihnen.

2. Als Bausätze
Unsere Bausätze enthalten alle Teile, die notwendig sind, um die beschriebene Erweiterung komplett aufzubauen. Sie brauchen die Bauteile nur noch gemäß der Anleitung in dem jeweiligen Heft zusammenzulöten und einzubauen.

3. Als Fertiggeräte
Die Fertiggeräte sind komplett aufgebaute und geprüfte Geräte. Sie brauchen die Erweiterung lediglich noch einzubauen.

Wichtiger Hinweis: Wir bemühen uns um eine umgehende Auslieferung Ihrer bestellten Hardware. Aber bis zum Eingang Ihrer Überweisung, der Auftragsabwicklung und der dazugehörenden Postwege vergehen mindestens 3 Wochen. Bitte haben Sie Verständnis, wenn aus diesen Gründen Ihre Hardware nicht sofort bei Ihnen eintrifft.

Unser Angebot

Angebot 1:

Expansion-Port Eprom-Platine mit 1 x 8 KByte Speicherplatz für 2732 bis 2764 Eproms.

Beschreibung in Ausgabe 10/85
Bestellnummer: HW 010 pro Stück **19,80***
Dieser Artikel wird nur als Fertiggerät angeboten.

Angebot 2:

Expansion-Port Eprom-Platine mit 2 x 8 KByte Speicherplatz für 2732 bis 2764 Eproms, mit Umschaltmöglichkeit.

Beschreibung in Ausgabe 10/85

Leerplatine
Bestellnummer: HW 020 pro Stück **24,80***

Bausatz mit allen Teilen:
Bestellnummer: HW 021 pro Stück **49,80***

Fertiggerät, getestet, wie beschrieben:
Bestellnummer: HW 022 pro Stück **59,80***

Angebot 3:

Eprom Trans – Die Speichererweiterung

ROM-Speichererweiterung zum Einbau in den C64, gleichzeitig Steckplatz für ein Original- oder ein alternatives Betriebssystem. Zwei Platinen in Epoxid-Harz-Ausführung wie in Ausgabe 10/85 beschrieben.

Leerplatine
Bestellnummer: HW 030 pro Stück **49,80***

Bausatz mit allen Teilen:
Bestellnummer: HW 031 pro Stück **119,80***

Eprom-Trans ist nicht als Fertiggerät erhältlich. Die Hardware-Erweiterungen aus früheren Ausgaben und die 40/80 Zeichen-Umschaltung für den C128 werden wir so bald als möglich in unser Angebot aufnehmen.

Angebot 4:

Super Kernal

Erweitertes Betriebssystem für den C 64 mit vielen neuen Funktionen inkl. Adaptersockel, einbaufertig in den C 64.
Beschreibung in Ausgabe 11/85

Version 1: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / RS232
Bestellnummer: HW 040

Version 2: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / Super Centronics Schnittstelle
Bestellnummer: HW 041

Version 3: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / Hypra Save
Bestellnummer: HW 042

Version 4: Enthält Hypra Load / DOS 5.1 / Funktionstasten / Hypra Save / Centronics klein
Bestellnummer: HW 043

Preis für jede Version pro Stück: **39,80***

* Alle Preise inklusive Mehrwertsteuer

Qualität & Service

- Die 64'er Hardware hat einen hohen Qualitätsstandard. Wir verwenden nur beste Epoxid-Harz-Platinen mit Lötstopp-Lack.
- Wir verwenden nur Präzisionssockel mit gedrehten Kontakten.
- Alle Platinen werden professionell gefertigt. Wenn notwendig mit doppelseitiger Beschichtung und Löt-Durchkontaktierungen.
- Jedes Gerät, das wir versenden, wurde auf Funktionstüchtigkeit geprüft.
- Wir sind auch nach dem Verkauf für Sie da. Neben der gesetzlichen Garantie bietet unser Service- und Fertigungspartner Ihnen Hilfe und Unterstützung an.

Unsere Garantie

Im Rahmen der Versand- und Lieferbedingungen unterliegen die Geräte einer Gewährleistungszeit von 6 Monaten ab Lieferung. Der Lieferung liegt eine Service-Karte bei, die Sie im Falle einer Beanstandung zusammen mit dem Gerät an die auf der Karte vermerkte Adresse schicken können. Die gleiche Karte verwenden Sie bitte bei Reparaturen nach der Garantiezeit.

Wie bestelle ich?

Alle Hardware-Erweiterungen, die Sie bestellen können, tragen einen Bestellverweis am Ende des Artikels im jeweiligen Heft. Falls Sie keinen Hinweis finden, hat sich der Autor dieser Erweiterung nicht dazu entschließen können, seine Entwicklung im Rahmen des Leserservice für eine Verbreitung freizugeben. Bitte verwenden Sie für Ihre Bestellung immer die beiliegende Postscheck-Zahlkarte oder einen Verrechnungsscheck. Sie erleichtern uns damit die Auftragsabwicklung und sparen sich Versandkosten.

64er-online.de
64er-online.net

Wie schon erwähnt, muß bei nicht aufgeführten Druckern (Typ 0) der Zeilenvorschub vor dem Start des Programms auf 7/72 inch anhand der Bedienungsanleitung verkleinert und nach dessen Ausführung wieder auf den normalen Wert rückgesetzt werden.

Unter Typ 3 fallen alle Drucker, bei denen die Umschaltung mit dem Befehl »ESC 1« (PRINT #fn,CHR$(27);chr$(49);) erfolgt.

Nachdem Sie die Eingaben beendet haben, wird der Poster-Maker als Maschinenprogramm auf Diskette gespeichert. Er muß mit Load »POSTER-MAKER«,8,1 geladen werden. Haben Sie den Poster-Maker mit dem Basic-Befehl

SYS 49152, adr, m, c1, c2, c3, c4

für Multicolour-Bilder beziehungsweise

SYS 49152, adr, m, c1, c2

für normalen Einzelpunktmodus gestartet, wird die gewünschte Grafik ausgedruckt. Zu beachten ist, daß bei der Eingabe der CHR$-Codes für die Druckzeichen die ASCII-Tabelle des Druckers, nicht die im Handbuch des C 64, gültig ist. Diese stimmt nur für Commodore-Drucker.

Ein CLOSE nach Beendigung des Ausdrucks ist nicht nötig. Jetzt muß nur noch die Druckerfahne an den gestrichelten Linien auseinandergeschnitten und die vier Streifen nebeneinandergeklebt werden.

(Thomas Wolf/ah)

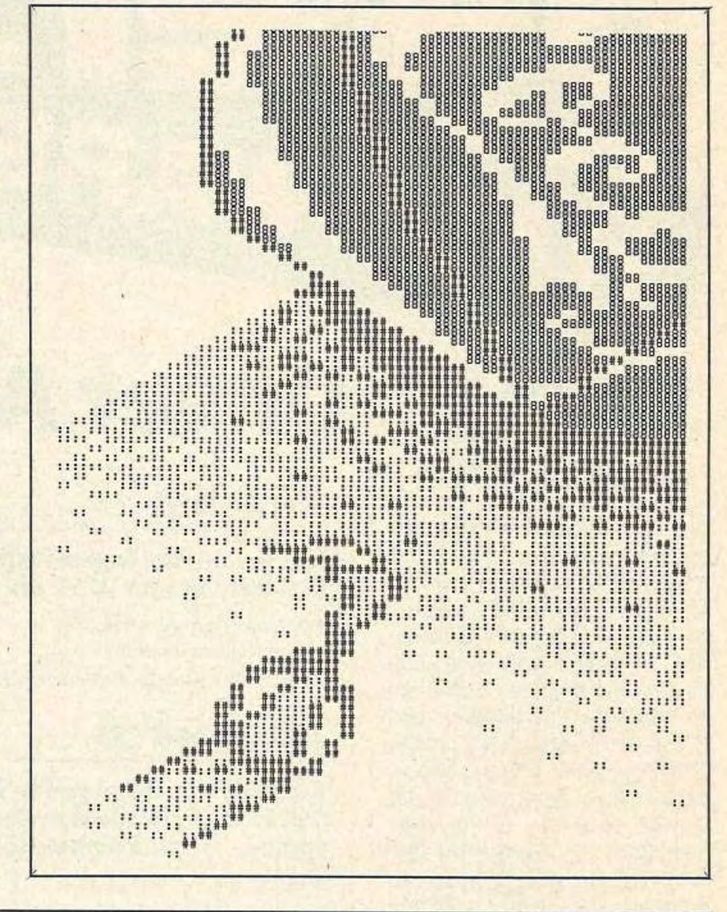


Bild 2. Damit man erkennt, wie »Poster-Maker« eine Grafik erstellt, hier ein Ausschnitt mit dem FX-80 ►

```
100 REM    ************************      <153>
110 REM    *                      *      <159>
120 REM    *    POSTER - MAKER    *      <027>
130 REM    *                      *      <179>
140 REM    *     THOMAS WOLF      *      <182>
150 REM    *    KIEFERNWEG 16     *      <031>
160 REM    *    6940 WEINHEIM     *      <164>
170 REM    *   TEL.06201/13551    *      <198>
180 REM    *                      *      <229>
190 REM    ************************      <243>
200 REM                                  <006>
210 PRINT"{CLR,DOWN}"TAB(8)"P O S T E R -
    M A K E R"                           <146>
220 PRINT TAB(8)"TTTTTTTTTTTTTTTTTTTTTTTT" <035>
230 FOR I=49152 TO 49609:READ A:S=S+A:POKE
     I,A:NEXT                            <131>
240 IF S<>54783 THEN PRINT TAB(9)"{4DOWN}F
    EHLER IN DATA-ZEILEN.":END           <253>
250 INPUT"{3DOWN,3SPACE}FILENUMMER........
    ..{2SPACE}1{3LEFT}";F                <204>
260 INPUT"{DOWN,3SPACE}SEKUNDAERADRESSE...
    .{2SPACE}0{3LEFT}";S                 <188>
270 PRINT"{DOWN,3SPACE}DRUCKERTYP.........
    .{2SPACE}1"                          <141>
280 PRINT"{DOWN,4SPACE}1 = MPS 801/VC 1525
    "                                    <112>
290 PRINT"{4SPACE}2 = MPS 802/VC 1526"   <154>
300 PRINT"{4SPACE}3 = EPSON UND KOMPATIBLE
    "                                    <248>
310 PRINT"{4SPACE}0 = SONSTIGE (VOR DEM ST
    ART ZEILEN-"                         <104>
320 PRINT"{17SPACE}VORSCHUB AENDERN.)"   <088>
330 PRINT"{7UP}"TAB(23);:INPUT D         <012>
340 POKE 49352,F:POKE 49357,S:POKE 49519,D <233>
350 FOR I=0 TO 3:POKE 828+I,PEEK(43+I):NEX
    T                                    <244>
360 POKE 43,0:POKE 44,192                <064>
370 POKE 45,202:POKE 46,193              <151>
380 SAVE"POSTER-MAKER",8                 <037>
390 POKE 43,PEEK(828):POKE 44,PEEK(829):PO
    KE 45,PEEK(830):POKE 46,PEEK(831)    <109>
400 PRINT"{CLR}OK.":END                  <172>
410 DATA 32,132,192,160,0,177,92,133,85,16
    5,2,208,20,162,8,6,85,169,0,105      <135>
420 DATA 0,168,185,172,193,32,210,255,202,
    208,240,240,27,162,4,169,0,6,85      <035>
430 DATA 105,0,6,85,144,2,9,2,168,185,172,
    193,32,210,255,32,210,255,202,208    <156>
440 DATA 231,160,0,32,7,193,144,192,32,110
    ,193,192,6,240,7,32,7,193,144,180    <044>
450 DATA 176,245,162,25,32,84,193,32,91,19
    3,32,7,193,144,165,173,111,193,201   <128>
460 DATA 1,144,22,201,2,208,8,169,36,72,32
    ,148,193,208,10,169,27,32,210,255    <088>
470 DATA 169,64,32,210,255,32,204,255,169,
    1,32,195,255,96,32,253,174,32,235    <198>
480 DATA 183,160,8,24,165,20,72,105,56,153
    ,192,193,200,165,21,72,105,1,153     <252>
490 DATA 192,193,160,0,169,2,133,171,202,1
    34,2,240,11,224,1,240,3,76,8,175     <206>
500 DATA 169,4,133,171,152,72,32,253,174,3
    2,158,183,104,168,138,153,172,193    <104>
510 DATA 200,196,171,208,237,169,1,72,162,
    4,160,0,32,136,193,104,170,32,201    <176>
520 DATA 255,173,111,193,201,2,144,28,201,
    3,208,13,169,27,32,210,255,169,49    <198>
530 DATA 32,210,255,76,249,192,169,18,72,3
    2,148,193,162,1,32,201,255,32,91     <214>
540 DATA 193,104,168,104,170,152,160,6,72,
    76,27,193,152,72,24,185,92,0,121     <044>
550 DATA 176,193,170,200,185,92,0,121,176,
    193,136,240,32,150,92,200,153,92     <177>
560 DATA 0,136,72,24,138,121,184,193,153,1
    92,193,200,185,92,0,121,184,193      <005>
570 DATA 153,192,193,104,136,136,136,208,2
    24,134,92,133,93,104,168,185,92      <006>
580 DATA 0,200,200,217,192,193,136,185,92,
    0,200,200,249,192,193,136,96,32      <099>
590 DATA 110,193,202,208,250,96,32,130,193
    ,162,80,169,45,32,210,255,202,208    <155>
600 DATA 250,162,25,32,84,193,96,169,1,201
    ,1,208,14,169,8,32,210,255,32,130    <056>
610 DATA 193,169,15,32,210,255,96,169,13,3
    2,210,255,96,32,186,255,169,0,32     <096>
620 DATA 189,255,32,192,255,96,169,6,162,4
    ,160,6,32,136,193,162,6,32,201,255   <047>
630 DATA 104,32,210,255,162,6,32,195,255,9
    6,32,32,32,32,8,0,1,0,64,1,80,0      <011>
640 DATA 0,0,80,0,8,0,64,31,0,0,0,0,0,0,0,
    0,0,0                                <076>
```

Listing zum Programm »Poster-Maker«. Bitte beachten Sie die Eingabehinweise auf Seite 7

Directory dreispaltig gedruckt

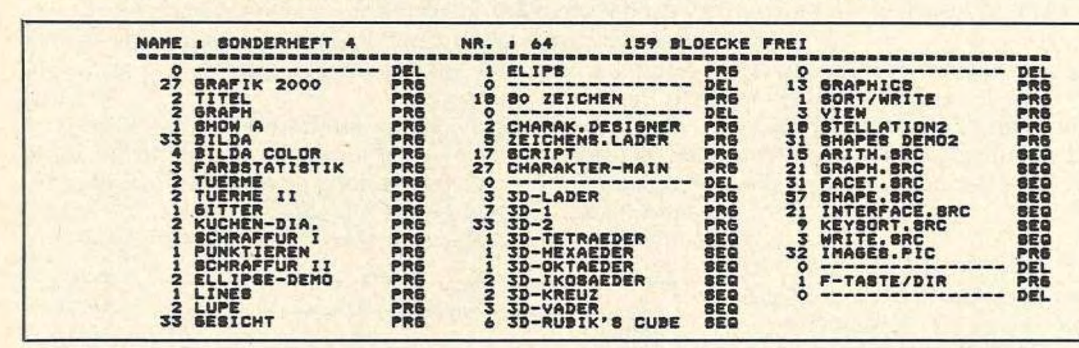
Besitzer eines Epson-kompatiblen Druckers können ihre Disketten-Directorys nun dreispaltig zum Aufkleben für die Diskettenhülle ausdrucken.

Es ist immer das gleiche: Disk rein – Disk raus. Man sucht ein bestimmtes Programm auf irgendeiner Diskette, und es vergeht geraume Zeit, bis man fündig wird. Hilfe soll dieses Programm bringen, das den Disketteninhalt in komprimierter Schrift so auf ein Papier bringt, daß man es anschließend nur noch auszuschneiden braucht und mit Kleber auf die Diskettenhülle aufpunktet (Bild 1).

152 Einträge, mehr passen auch nicht auf eine Diskseite, lassen sich damit ausdrucken. Dieses Programm läuft auf Epson- und Star SG 10/15-Druckern mit Görlitz-Interface und auch über die Centronics-Schnittstelle, zum Beispiel Eyssele-Schnittstelle.

Für das Wiesemann- und Data-Becker-Interface ist Zeile 470 in »OPEN 1,4,1« umzuändern. Die Bedienung des Programms (Listing 1) erklärt sich von alleine. Der Drucker ist vorher einzuschalten, ansonsten kommt in Zeile 490 eine Fehlermeldung.

(Klaus Grabietz/bs)



```
NAME : SONDERHEFT 4          NR. : 64       159 BLOECKE FREI
=============================================================================
  0 ----------------  DEL     1 ELIPS              PRG     0 ----------------  DEL
 27 GRAFIK 2000       PRG     0 ----------------   DEL    13 GRAPHICS          PRG
  2 TITEL             PRG    18 80 ZEICHEN         PRG     1 SORT/WRITE        PRG
  2 GRAPH             PRG     0 ----------------   DEL     3 VIEW              PRG
  1 SHOW A            PRG     2 CHARAK.DESIGNER    PRG    18 STELLATION2       PRG
 33 BILDA             PRG     5 ZEICHENS.LADER     PRG    31 SHAPES DEMO2      PRG
  4 BILDA COLOR       PRG    17 SCRIPT             PRG    15 ARITH.SRC         SEQ
  3 FARBSTATISTIK     PRG    27 CHARAKTER-MAIN     PRG    21 GRAPH.SRC         SEQ
  2 TUERME            PRG     0 ----------------   DEL    31 FACET.SRC         SEQ
  2 TUERME II         PRG     3 3D-LADER           PRG    57 SHAPE.SRC         SEQ
  1 GITTER            PRG     7 3D-1               PRG    21 INTERFACE.SRC     SEQ
  2 KUCHEN-DIA.       PRG    33 3D-2               PRG     9 KEYSORT.SRC       SEQ
  1 SCHRAFFUR I       PRG     1 3D-TETRAEDER       SEQ     3 WRITE.SRC         SEQ
  1 PUNKTIEREN        PRG     1 3D-HEXAEDER        SEQ    32 IMAGES.PIC        PRG
  1 SCHRAFFUR II      PRG     1 3D-OKTAEDER        SEQ     0 ----------------  DEL
  2 ELLIPSE-DEMO      PRG     2 3D-IKOSAEDER       SEQ     1 F-TASTE/DIR       PRG
  1 LINES             PRG     2 3D-KREUZ           SEQ     0 ----------------  DEL
  2 LUPE              PRG     3 3D-VADER           SEQ
 33 GESICHT           PRG     6 3D-RUBIK'S CUBE    SEQ
```

Bild 1. Ein typischer Beispiel-Ausdruck

```
10 POKE 53280,0:POKE 53281,1:PRINT CHR$(30
   )CHR$(14)CHR$(147)                        <224>
20 PRINT"{18DOWN,3SPACE}DER DISKETTENINHAL
   T WIRD IN KLEIN-"                         <135>
30 PRINT                                     <132>
40 PRINT" SCHRIFT AUF EINEM DRUCKER AUSGEG
   EBEN."                                    <079>
50 PRINT"{3DOWN,4SPACE}BITTE "CHR$(18)"'DR
   UCKER'"CHR$(146)" EINSCHALTEN             <091>
60 PRINT                                     <162>
70 PRINT"{7SPACE}UND TASTE DRUECKEN !        <175>
80 POKE 198,0                                <244>
90 GET A$:M=M+1:IF M=10 THEN M=0             <071>
100 PRINT MID$("{RVSON,RVOFF}",M/5+1,1);     <072>
110 PRINT"{HOME,5DOWN}KLAUS GRABIETZ,AM WE
    INBERG14,3108 WINSEN"                    <049>
120 PRINT"{RVSON,HOME,2DOWN,13SPACE}FILE P
    RINTER{15SPACE}"                         <249>
130 IF A$=""GOTO 90                          <166>
140 PRINT"{17DOWN,7SPACE}NAECHSTE DISKETTE
     - "CHR$(18)"'F7'"                       <032>
150 PRINT                                    <252>
160 PRINT"{20SPACE}ENDE - "CHR$(18)"'F1'"    <008>
170 POKE 198,0                               <078>
180 GET C$:IF C$=CHR$(136)THEN 210           <051>
190 IF C$=CHR$(133)THEN PRINT"{CLR}":END     <229>
200 GOTO 180                                 <000>
210 DIM F$(152)                              <207>
220 DIM G$(152)                              <225>
230 PRINT CHR$(147)CHR$(31)CHR$(142)         <178>
240 OPEN 9,8,0,"$"                           <052>
250 GET#9,A$,B$:N0=0                         <210>
260 GET#9,A$,B$:Z0=0:X$=""                   <020>
270 IF ST<>0 THEN 350                        <121>
280 GET#9,A$,B$                              <115>
290 N$=CHR$(0)                               <236>
300 G$=RIGHT$("{6SPACE}"+STR$(ASC(A$+N$)+A
    SC(B$+N$)*256),6)                        <240>
310 GET#9,A$:IF A$=CHR$(34)THEN Z0=SGN(Z0+
    1)                                       <157>
320 IF A$=""THEN N0=N0+1:N1=N0-1:GOSUB 360
    :GOTO 260                                <055>
330 IF Z0=1 THEN G$=G$+A$                    <080>
340 GOTO 310                                 <046>
350 CLOSE 9:GOTO 470                         <087>
360 IF N0=1 THEN 440                         <222>
370 IF LEN(G$)=6 THEN 460                    <141>
380 AA$="":FOR I=8 TO 24:C$=MID$(G$,I,1):I
    F C$=CHR$(34)THEN I=24:GOTO 400          <061>
390 AA$=AA$+C$                               <198>
400 NEXT I:G$(N1)=LEFT$(AA$+"{18SPACE}",18
    )+LEFT$(G$,6)+"{2SPACE}"+ID$             <129>
410 H1$=MID$(G$(N1),22,3):H2$=LEFT$(G$(N1)
    ,16)                                     <161>
420 H$=H1$+" "+H2$+" "+MID$(G$,26,3):PRINT
    "{7SPACE}"H$:A=A+1:F$(A)=H$:PRINT CHR$
    (142)                                    <144>
430 RETURN                                   <234>
440 CC$=RIGHT$(G$,24):ID$=MID$(CC$,20,2):P
    RINT"{7SPACE,RVSON}";CC$:RETURN          <043>
450 PRINT                                    <042>
460 PRINT"{4SPACE}";G$;" BLOECKE FREI.":CC
    $=CC$+G$+" BLOECKE FREI.{2SPACE}"+DA$:
    RETURN                                   <086>
470 OPEN 1,4                                 <039>
480 PRINT#1:CMD 1 :PRINT CHR$(27)CHR$(15);
    :PRINT CHR$(27)"S"CHR$(0);               <182>
490 PRINT#1,CHR$(27)"A"CHR$(5);"NAME : ";M
    ID$(CC$,2,16);                           <008>
500 PRINT#1,"{5SPACE}NR. : ";ID$;            <104>
510 PRINT#1,"{6SPACE}"MID$(CC$,28,16)        <039>
520 PRINT#1,"=============================
    ================================="       <228>
530 PRINT#1,"================="              <093>
540 IF A/3<>INT(A/3)AND A-3*INT(A/3)=2 THE
    N A=A+3:GOTO 560                         <085>
550 IF A/3<>INT(A/3)AND A-3*INT(A/3)=1 THE
    N A=A+3:GOTO 560                         <091>
560 FOR X=1 TO INT(A/3)                      <219>
570 PRINT#1," "F$(X);"{3SPACE}";             <020>
580 PRINT#1,F$(X+INT(A/3));"{3SPACE}";       <178>
590 PRINT#1,F$(X+2*INT(A/3))                 <109>
600 NEXT:GOTO 620                            <057>
610 PRINT#1,MID$(CC$,28,16)                  <123>
620 CLOSE 1                                  <123>
630 PRINT CHR$(30)CHR$(14)CHR$(147):RUN 14
    0                                        <088>
```

© 64'er

Das Listing zu »File Printer«. Bitte beachten Sie die Eingabehinweise auf Seite 6.

Auto-Save

Wenn Ihnen beim Abtippen eines Programms schon mal der Strom ausgefallen ist, dann ist das folgende Programm genau das richtige für Sie.

Die abgedruckten Listings in den Fachzeitschriften sind ja eine willkommene Programmquelle. Allerdings ist das Abtippen der oft sehr langen Programme eine sehr mühselige Angelegenheit. Besonders groß wird der Ärger, wenn ein Stromausfall beziehungsweise ein Hardware-Defekt stundenlange Tipparbeit zunichte machen. Abhilfe schafft hier nur das umständliche Zwischenspeichern in gewissen Abständen, doch das wird oft vergessen oder aus Bequemlichkeit unterlassen.

Das hier vorgestellte Programm nimmt nun dem geplagten Computer-Benutzer diese Arbeit ab. Es speichert automatisch das bisher Getippte in Intervallen oder auch auf Knopfdruck ab. Bitte tippen Sie vor der Benutzung Listing 1 mit dem MSE und Listing 2 mit dem Checksummer ab. Vergessen Sie das Speichern nicht!

Hinweise zur Benutzung:

Geladen wird das Programm mit LOAD"AUTO-SAVE",8 und anschließendem RUN. Der Maschinensprachteil wird nachgeladen und dann der Name des einzutippenden Programms erfragt. Keine beziehungsweise Eingabe von Leerzeichen wird abgefangen.

Bei der Eingabe des Speicher-Intervalls ist folgendes zu beachten: Die Intervalle werden nicht durch die Zeit, sondern durch die Speicherbelegung gesteuert. Dies geschieht durch eine Abfrage der Speicherstelle 46. Diese enthält das High-Byte des Zeigers auf den Variablen-Anfang (=Programmende). Gibt man also als Intervall zum Beispiel 4 ein, so wird immer dann, wenn 1 KByte Programmtext neu im Speicher stehen, zwischengespeichert. Um außerdem besonders schwierige Programmteile, zum Beispiel Bildschirmmasken oder mathematische Formeln, sofort zu speichern, gibt es die Möglichkeit, per Knopfdruck ein Speichern auszulösen.

Dies geschieht durch Drücken der @-Taste (Klammeraffe) und anschließendem RETURN. Diese Funktion kann mit POKE 49177,ASC("taste") leicht auf jede andere Taste verlegt werden.

Beendet werden kann das Programm durch RUN STOP/RESTORE oder einen Reset, neu wird es gestartet mit SYS 49152.

Erläuterungen zum Programm: »SAVE.OBJ«:

Die Gliederung des Maschinenprogramms läßt sich am besten anhand einer Tabelle durchführen:

Speicherbereich hexadezimal	Belegung
$C000-C014	INIT
$C015-C02A	BEFEHL
$C02B-C037	AUTO
$C038-C0A6	SAVE
$C0A7-C0D2	Textmeldung
$C0D3	Sollwert
$C0D4	Intervall
$C0D5	Namenlänge
$C0D6	Namenlänge+ 2
$C0D7-C0D8	"S:"
$C0D9	Programmname

INIT: Zwei Vektoren werden umgeändert und damit das Maschinenprogramm aktiviert.

BEFEHL: Dieser Teil wird über den Vektor "Neuen Basic-Befehl ausführen" (dez. 776-777) angesprungen. Er prüft, ob der Klammeraffe beziehungsweise die festgelegte Taste gedrückt wurde. Falls ja, wird zu SAVE verzweigt, ansonsten wieder in die Interpreterschleife.

AUTO: Diese Routine wurde über den "Zeiger auf Tastatur-Dekodiertabelle" (dez. 655-656) eingebunden. Bei jedem Tastendruck wird der Inhalt der Speicherstelle 46 mit dem Sollwert (siehe oben) verglichen und bei Gleichheit wiederum SAVE angesprungen.

SAVE: Der eigentliche Hauptteil des Programms. Er wird von den oben genannten Routinen angesprungen und hat mehrere Aufgaben:
- »Sollwert« um »Intervall« erhöhen
- bisherigen Programmteil löschen
- neuen Programmteil speichern
- Textmeldung ausgeben.

Um den »Replace«-Befehl mittels Klammeraffen, der bekanntlich seine Tücken hat, zu umgehen, wurde der etwas umständlichere Weg gewählt, vor dem Speichern einen »Scratch«-Befehl an das Laufwerk zu senden und somit das alte Programm zu löschen.

Die Meldung »Saving« des Interpreters wird unterdrückt und statt dessen der Text »Programm wird abgespeichert« ausgegeben. Wem diese Meldung nicht gefällt, der kann sie leicht ändern, indem er sie im ASCII-Format in den oben genannten Speicherbereich schreibt. (Manfred Lins/bs)

```
1 REM  **********************************  <051>
2 REM  *                                *  <051>
3 REM  *           AUTO - SAVE          *  <047>
4 REM  *                                *  <053>
5 REM  *      1985 BY MANFRED LINS      *  <120>
6 REM  *             REITACKER 1        *  <209>
7 REM  *        6492 ZUENTERSBACH       *  <080>
8 REM  *                                *  <057>
9 REM  **********************************  <059>
10 :                                        <242>
11 :                                        <243>
12 REM  MASCHINEN-PROGRAMM NACHLADEN        <181>
13 IF R=0 THEN R=1:LOAD"SAVE-OBJ",8,1       <058>
14 :                                        <246>
16 PRINT CHR$(147)CHR$(18)"{13SPACE}AUTO -
    SAVE{16SPACE}"                          <014>
19 PRINT:PRINT:PRINT                        <144>
21 :                                        <253>
22 REM  NAME EINGEBEN                       <001>
23 PRINT"{2SPACE}PROGRAMMNAMEN (MAX. 16 ZE
   ICHEN) EIN-"                             <102>
24 PRINT"{2SPACE}GEBEN :"                   <039>
25 :                                        <001>
26 S=10:Z=6:GOSUB 62                        <235>
27 IF A$="" OR LEFT$(A$,1)=" " THEN 26      <186>
28 :                                        <004>
29 REM  NAMEN UND LAENGE ABSPEICHERN        <239>
30 N$="S:"+LEFT$(A$,16):L=LEN(N$)           <010>
31 POKE 49365,L-2:POKE 49366,L              <027>
32 :                                        <116>
33 FOR I=1 TO L:POKE 49366+I,ASC(MID$(N$,I
   ,1)):NEXT I                              <007>
36 PRINT:PRINT                              <244>
39 REM  INTERVALL EINGEBEN                  <005>
40 PRINT"{2SPACE}ABSPEICHER-INTERVALL (256
    BYTE = 1)"                              <120>
41 PRINT"{2SPACE}EINGEBEN :"                <138>
43 S=13:Z=10:GOSUB 62                       <016>
46 REM  INTERVALL ABSPEICHERN               <184>
47 I=VAL(A$):IF I<1 THEN I=1                <195>
48 POKE 49363,8+I:POKE 49364,I              <128>
51 REM MASCHINENPROGRAMM INITIALISIEREN     <201>
52 SYS 49152                                <110>
53 :                                        <029>
54 REM  LADEPROGRAMM LOESCHEN               <191>
55 NEW:END                                  <154>
60 REM  UNTERPROGRAMM STRINGEINGABE         <083>
62 PRINT CHR$(144):OPEN 1,0                 <029>
63 POKE 214,Z:POKE 211,S:SYS 58640          <036>
64 INPUT#1,A$                               <206>
65 CLOSE 1:PRINT CHR$(154):RETURN           <079>
```

Listing 2. »AUTO-SAVE«: Sicherheit bei Stromausfall

```
programm : save-obj             c000 c0d3
-----------------------------------------

c000 : a2 15 a0 c0 8e 08 03 8c   bb
c008 : 09 03 a2 2b a0 c0 8e 8f   0a
c010 : 02 8c 90 02 60 20 73 00   92
c018 : c9 40 f0 06 20 79 00 4c   65
c020 : e7 a7 20 73 00 20 38 c0   b5
c028 : 4c 7b a4 a5 2e cd d3 c0   32
c030 : d0 03 20 38 c0 4c 48 eb   f8
c038 : a9 01 a2 08 a0 6f 20 ba   87
c040 : ff a9 00 20 bd ff 20 c0   f6
c048 : ff a9 08 20 b1 ff a9 6f   c3
c050 : 20 93 ff a2 00 bd d7 c0   5d
c058 : 20 a8 ff e8 8a cd d6 c0   dd
c060 : d0 f3 a9 08 20 ae ff a9   60
c068 : 01 20 c3 ff ad d3 c0 18   17
c070 : 6d d4 c0 8d d3 c0 a2 00   f7
c078 : bd a7 c0 20 d2 ff e8 8a   23
c080 : c9 2c d0 f4 a9 00 a2 08   67
c088 : 20 ba ff a2 d9 a0 c0 ad   5a
c090 : d5 c0 20 bd ff a9 00 85   dd
c098 : 9d a9 2b a6 2d a4 2e 20   9a
c0a0 : d8 ff a9 80 85 9d 60 11   db
c0a8 : 20 20 20 20 20 12 20 50   98
c0b0 : 52 4f 47 52 41 4d 4d 20   ba
c0b8 : 57 49 52 44 20 41 42 47   75
c0c0 : 45 53 50 45 49 43 48 45   c6
c0c8 : 52 54 20 92 20 20 20 20   63
c0d0 : 20 20 11 85 3b 85 2d 4c   23
```

Listing 1. »SAVE.OBJ« wird von Listing 2 »AUTO-SAVE« nachgeladen. Bitte beachten Sie die Eingabehinweise auf Seite 7.

Ziffern und Zeiger auf dem C 64

Digitaluhren sind out. Um dem Trend der Zeit zu folgen, haben wir für Sie eine »Analog-Uhr« auf dem C 64 installiert.

Es sind schon etliche Versionen von Digitaluhren auf dem C 64 veröffentlicht worden. Zur Abwechslung haben wir diesmal eine Interrupt-gesteuerte Analoguhr für Sie. Das Programm blendet ein klassisches Ziffernblatt mit Zeigern in die rechte obere Ecke des Bildschirms ein.

Hinweise zum Eintippen. Geben Sie zuerst das Maschinenprogramm »UHR PRG.« (Listing 1) mit dem MSE ein, und speichern Sie es. Dann tippen Sie »UHR SPRITEGEN« (Listing 2) mit dem Checksummer ab und speichern es ebenfalls. Starten Sie »UHR SPRITEGEN«. Dieses lädt zuerst das Maschinenprogramm und berechnet danach die Sprite-Daten für die Uhr und die Zeiger. Nach etwa vier Minuten werden die Spritedaten inklusive der Interruptroutine auf Diskette gespeichert. Das Programm wird dann mit LOAD "ANALOGUHR",8,1 geladen.

Zum Programm. Die Uhr wird durch »SYS 40000,hh, mm,ss« eingeschaltet. Dabei bedeuten hh, mm, ss jeweils Stunden, Minuten und Sekunden mit je zwei Ziffern (Zum Beispiel »SYS 40000,15,05,00« stellt die Uhr auf 15:05,00). Der Bildschirmspeicher liegt jetzt von 35840 bis 36839 und der Basic-Speicher wird auf etwa 33 000 Bytes begrenzt. Durch RUN/STOP-RESTORE wird die Uhr abgeschaltet und der Bildschirm liegt wieder bei 1024. Das Ziffernblatt und die drei Zeiger bestehen aus Sprites, die von der neuen Interrupt-Routine gesteuert werden. Diese Routine liest bei jedem Interrupt die »time-of-day«-Register der CIA 1, berechnet für jeden Zeiger die Bildschirmposition und setzt die Spritpointer. Dann wird zum normalen Interrupt des Betriebssystems gesprungen.

Programmbeschreibung: Das Basic-Programm »UHR SPRITEGEN« besteht hauptsächlich aus drei Schleifen. In den Zeilen 160–260 werden die Zeigersprites erzeugt. Die IF-Zeilen bestimmen die Startkoordinaten der Linien. Durch READ in Zeile 220 werden die Endkoordinaten aus dem ersten Datenblock gelesen. Die aufgerufenen Unterprogramme löschen das zu bearbeitenden Sprite (460), ziehen eine Linie im Sprite (370) und übertragen das Sprite in den endgültigen Speicher unter dem Basic-ROM (480). In den Zeilen 270–290 werden die Daten für die vier Sprites, die das Zifferblatt darstellen, gelesen und gePOKEt. Zeile 300 liest die Daten für das Maschinenprogramm, das das Programm »ANALOGUHR« speichert. Dieses wird in Zeile 310 aufgerufen. Das Unterprogramm Zeile 340–360 setzt einen Punkt im bearbeiteten Sprite, in Zeile 380–440 werden die Punkte für eine Linie (X1, Y1, X2, Y2) berechnet und gesetzt. Zeile 460 löscht das Arbeitssprite. In Zeile 480 werden die Spritedaten in den Speicherbereich, entsprechend der Variablen »W«, die in Zeile 240 berechnet wird, übertragen. Die Datazeilen enthalten die X2,Y2-Koordinaten der Zeiger, die Daten für das Zifferblatt und für die Speicherroutine.

Das Maschinenprogramm »UHR PRG.« arbeitet folgendermaßen: nach SYS 40000 wird zuerst die Uhr in CIA I gestellt, der Bildschirm verschoben und die Spriteregister des VIC initialisiert. Dann wird der Basic-Speicher begrenzt und der NMI- und IRQ-Vektor verbogen. Anschließend wird zum Basic-Warmstart gesprungen. Die neue NMI-Routine setzt nur den Bildschirm wieder zurück, dann wird zum »Restore« des Betriebssystems gesprungen.

```
programm : uhr prg.             9c40 9e4c
-----------------------------------------

9c40 : ad 0f dc 29 7f 8d 0f dc   2b
9c48 : ad 0e dc 09 80 8d 0e dc   bb
9c50 : a2 03 d0 03 20 73 00 20   e6
9c58 : fd ae 38 e9 30 0a 0a 0a   87
9c60 : 0a 85 fd 20 73 00 38 e9   9c
9c68 : 30 05 fd c9 60 90 03 4c   03
9c70 : 48 b2 e0 03 d0 0d c9 24   8f
9c78 : b0 f5 c9 12 90 05 18 f8   5b
9c80 : 69 68 d8 9d 08 dc ca d0   3b
9c88 : cb 8e 08 dc a9 34 8d 18   db
9c90 : d0 a9 95 8d 00 dd a9 8c   fb
9c98 : 8d 88 02 a9 ff 8d 15 d0   81
9ca0 : 8d 10 d0 8e 29 d0 a9 01   fd
9ca8 : 8d 28 d0 8d 2a d0 8d 2b   e5
9cb0 : d0 8d 2c d0 8d 2d d0 8d   0d
9cb8 : 27 d0 a9 8c 85 34 85 38   c4
9cc0 : 78 a9 21 8d 14 03 a9 9d   42
9cc8 : 8d 15 03 a9 19 8d 18 03   3a
9cd0 : a9 9d 8d 19 03 68 68 58   94
9cd8 : a9 22 8d 06 d0 8d 0a d0   fa
9ce0 : a9 3a 8d 0c d0 8d 08 d0   c6
9ce8 : a9 38 8d 07 d0 8d 09 d0   31
9cf0 : a9 4d 8d 0b d0 8d 0d d0   54
9cf8 : a9 d4 8d fb 8f a9 d5 8d   a7
9d00 : fc 8f a9 d6 8d fd 8f a9   63
9d08 : d7 8d fe 8f a9 93 20 d2   b5
9d10 : ff a9 80 8d 00 a0 4c 74   d5
9d18 : a4 a9 04 8d 88 02 4c 66   da
9d20 : fe 78 ad 0b dc 29 1f 20   fb
9d28 : 30 9e aa 0a 18 69 bc 85   5e
9d30 : fd 8a c9 09 90 0e ad 4a   cb
9d38 : 9e 8d 00 d0 ad 4b 9e 8d   82
9d40 : 01 d0 d0 30 c9 06 90 0e   0f
9d48 : ad 48 9e 8d 00 d0 ad 49   42
9d50 : 9e 8d 01 d0 d0 1e c9 03   3a
9d58 : 90 0e ad 46 9e 8d 00 d0   1b
9d60 : ad 47 9e 8d 01 d0 d0 0c   fc
9d68 : ad 44 9e 8d 00 d0 ad 45   58
9d70 : 9e 8d 01 d0 ad 0a dc 20   0e
9d78 : 30 9e aa 09 80 8d f9 8f   3f
9d80 : 8a c9 2d 90 15 ad 4a 9e   71
9d88 : 8d 02 d0 ad 4b 9e 8d 03   e6
9d90 : d0 a9 01 05 fd 8d f8 8f   65
9d98 : d0 41 c9 1e 90 15 ad 48   38
9da0 : 9e 8d 02 d0 ad 49 9e 8d   5a
9da8 : 03 d0 a9 01 05 fd 8d f8   06
9db0 : 8f d0 28 c9 0f 90 13 a5   f8
9db8 : fd 8d f8 8f ad 46 9e 8d   4e
9dc0 : 02 d0 ad 47 9e 8d 03 d0   83
9dc8 : d0 11 ad 44 9e 8d 02 d0   15
9dd0 : ad 45 9e 8d 03 d0 a5 fd   c2
9dd8 : 8d f8 8f ad 09 dc 20 30   d3
9de0 : 9e aa 09 80 8d fa 8f 8a   2a
9de8 : c9 2d 90 0e ad 4a 9e 8d   f0
9df0 : 04 d0 ad 4b 9e 8d 05 d0   3d
9df8 : d0 30 c9 1e 90 0e ad 48   d7
9e00 : 9e 8d 04 d0 ad 49 9e 8d   3b
9e08 : 05 d0 d0 1e c9 0f 90 0e   e1
9e10 : ad 46 9e 8d 04 d0 ad 47   46
9e18 : 9e 8d 05 d0 d0 0c ad 44   85
9e20 : 9e 8d 04 d0 ad 45 9e 8d   3b
9e28 : 05 d0 ad 08 dc 4c 31 ea   cc
9e30 : aa 4a 4a 4a 4a a8 8a 29   42
9e38 : 0f c0 00 f0 06 18 69 0a   a0
9e40 : 88 d0 fa 60 36 39 36 4d   9c
9e48 : 22 4d 22 39 01 ff 34 01   a3
```

Listing 1. Das Maschinenprogramm »UHR PRG.« Bitte mit dem MSE eingeben.

```
100 REM   *** ANALOGUHR ***                     <038>
110 REM   MARC MICHAELIS                        <188>
120 REM   PFARRHOEHE 11                         <132>
130 REM   8507 OBERASBACH                       <035>
140 REM                                         <202>
150 IF A=0 THEN A=1:LOAD"UHR PRG.",8,1          <165>
160 FOR J=0 TO 83                               <161>
170 IF J=0 OR J=60 THEN X1=0:Y1=20              <164>
180 IF J=15 OR J=66 THEN X1=0:Y1=0              <079>
190 IF J=30 OR J=72 THEN X1=20:Y1=0             <050>
200 IF J=45 OR J=78 THEN X1=20:Y1=20            <101>
210 GOSUB 460                                   <018>
220 READ X2,Y2                                  <211>
230 GOSUB 370                                   <046>
240 W=40960+64*J                                <165>
250 GOSUB 480                                   <090>
260 NEXT                                        <016>
270 FOR J=46336 TO 46591                        <000>
280 READ A:POKE J,A                             <192>
290 NEXT                                        <046>
300 FOR J=832 TO 880:READ A:POKE J,A:NEXT       <234>
310 SYS 832                                     <236>
320 END                                         <068>
330 REM PLOT                                    <206>
340 Q=Y*3+INT(X/8)+832:Q1=2↑(7-(X-INT(X/8)
    *8))                                        <156>
350 POKE Q,PEEK(Q)OR Q1                         <108>
360 RETURN                                      <164>
370 REM LINIE ZIEHEN (X1,Y1,X2,Y2)              <066>
380 IF ABS(X2-X1)<5 THEN 420                    <231>
390 FOR I=X1 TO X2 STEP(X2-X1)/20               <253>
400 Y=INT((Y2-Y1)/(X2-X1)*(I-X1)+Y1)            <251>
410 X=INT(I):GOSUB 330:NEXT:RETURN              <039>
420 FOR I=Y1 TO Y2 STEP(Y2-Y1)/20               <204>
430 X=INT(((X2-X1)/(Y2-Y1)*(I-Y1)+X1))          <081>
440 Y=INT(I):GOSUB 330:NEXT:RETURN              <073>
450 REM LOESCHEN SPRITE                         <042>
460 FOR I=832 TO 896:POKE I,0:NEXT:RETURN       <066>
470 REM UEBERTRAGEN SPRITE                      <133>
480 FOR I=0 TO 63:POKE W+I,PEEK(832+I):NEX
    T:RETURN                                    <004>
490 REM DATA FUER ZEIGER ( X UND Y )            <074>
500 DATA 0,0,2,0,4,1,6,1,8,2,10,3,12,4,13,
    5,15,7,16,8,17,10,18,12,19,14,19,16         <123>
510 DATA 20,18                                  <143>
520 DATA 20,0,20,2,19,4,19,6,18,8,17,10,16
    ,12,15,13,13,15,12,16,10,17,8,18,6,19       <243>
530 DATA 4,19,2,20                              <228>
540 DATA 20,20,18,20,16,19,14,19,12,18,10,
    17,8,16,7,15,5,13,4,12,3,10,2,8,1,6         <142>
550 DATA 1,4,0,2                                <188>
560 DATA 0,20,0,18,1,16,1,14,2,12,3,10,4,8
    ,5,7,7,5,8,4,10,3,12,2,14,1,16,1,18,0       <141>
570 DATA 0,10,3,10,5,11,7,13,9,15,10,17         <078>
580 DATA 10,0,10,3,9,5,7,7,5,9,3,10             <209>
590 DATA 20,10,17,10,15,9,13,7,11,5,10,3        <214>
600 DATA 10,20,10,17,11,15,13,13,15,11,17,
    10                                          <154>
610 REM ZIFFERBLATT SPRITES                      <117>
620 DATA 0,0,255,0,7,8,0,24,8,0,32,0,0,224
    ,85,1,16,37,2,0,85,4,84,0,4,36,0,8          <075>
630 DATA 84,0,8,0,0,24,0,0,37,64,0,32,128,
    0,65,64,0,64,0,0,64,0,0,128,0,0,128         <142>
640 DATA 0,0,128,0,0,138,128,0,239,128,0,0
    ,112,0,0,12,0,0,2,0,0,3,128,0,4,64          <050>
650 DATA 0,0,32,0,16,16,0,16,8,0,16,4,0,0,
    4,0,0,6,0,0,169,0,0,161,0,0,160,128         <074>
660 DATA 0,0,128,0,0,128,0,0,64,0,0,64,0,0
    ,64,0,84,64,200,233,0,8,138,128,0           <230>
670 DATA 128,0,0,128,0,0,128,0,0,85,80,0,8
    5,80,0,73,80,0,32,0,0,36,0,0,24,0           <198>
680 DATA 0,8,170,0,8,170,0,4,74,0,2,0,42,1
    ,16,42,0,224,18,0,32,0,0,24,8,0,7           <035>
690 DATA 8,0,0,255,141,0,85,192,0,84,64,0,
    0,64,0,0,64,0,0,64,2,160,128,2,160          <068>
700 DATA 128,2,64,128,0,1,0,0,9,0,0,6,0,40
    ,4,0,40,4,0,16,24,0,0,32,0,4,64,0           <104>
710 DATA 3,128,0,2,0,0,12,0,0,112,0,0,128,
    0,0,0                                       <048>
720 REM DATEN FUER ABSPEICHERROUTINE            <032>
730 DATA 169,54,133,1,162,8,32,186,255,169
    ,9,162,104,160,3,32,189,255,169,64          <168>
740 DATA 133,251,169,156,133,252,169,251,1
    62,0,160,182,32,216,255,169,55,133          <008>
750 DATA 1,96,65,78,65,76,79,71,85,72,82        <088>
```

© 64'er

Listing 2. Das Basic-Programm »UHR SPRITEGEN« erzeugt die nötigen Sprites der Uhr, verbindet diese mit dem Maschinenprogramm und speichert beides als »ANALOGUHR« auf Diskette

Das eigentliche Steuerprogramm ist eine neue Interruptroutine. Zuerst wird die Stunde gelesen, von BCD in binär umgewandelt, und daraus die vorläufige Spritenummer für den Stundenzeiger berechnet. Außerdem wird die Spriteposition des Stundenzeigers berechnet (Stunde < 3 ≙ rechtes oberes Viertel, Stunde < 6 ≙ rechtes unteres Viertel etc.). Dann werden die Minuten gelesen, umgerechnet und die Position des Minutenzeigers berechnet. Wenn die Minuten größer oder gleich 30 sind, wird der Stundenzeiger um eine halbe Stunde vorgerückt. Schließlich wird noch der Sekundenzeiger nach der gleichen Methode gesetzt. Bevor zur IRQ-Routine des Betriebssystems gesprungen wird, muß noch das Zehntelsekunden-Register gelesen werden, um die Uhr wieder freizugeben.

(M. Michaelis/og)

Löschen ohne Verluste

Ein dimensioniertes Feld läßt sich normalerweise nicht ohne den Verlust aller Variablenwerte aus dem Speicher entfernen. Dieses Programm macht's möglich.

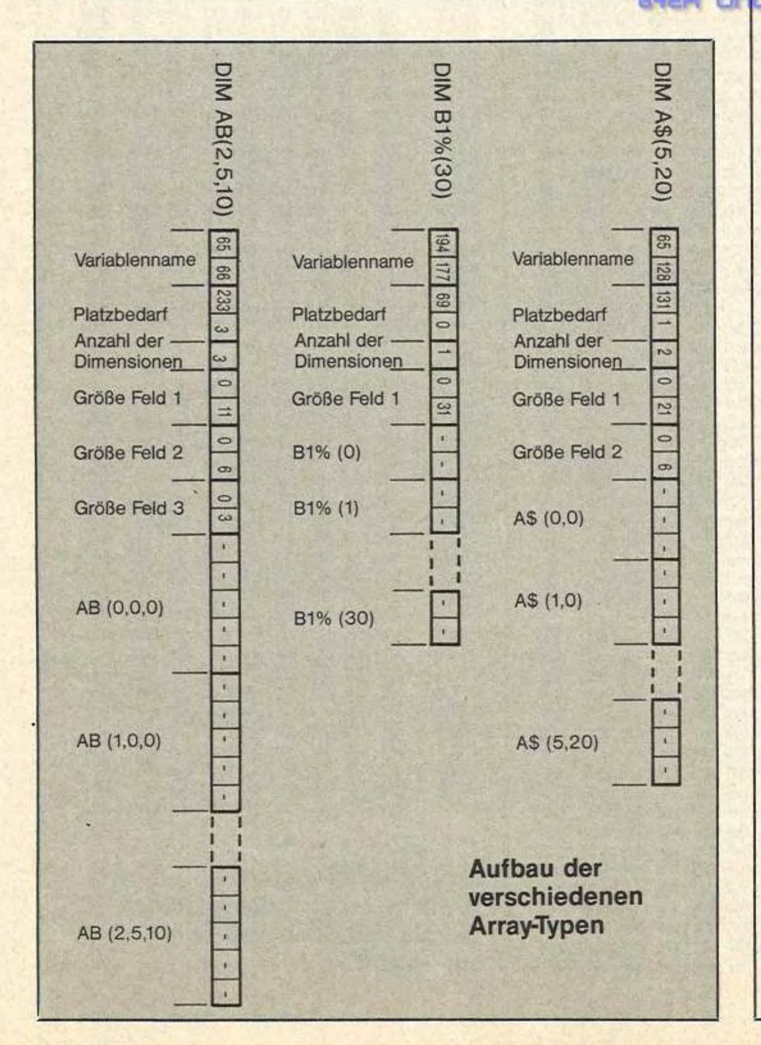
Ein einmal angelegtes Variablenfeld läßt sich beim Commodore 64 nicht wieder löschen; es sei denn, man benutzt den Befehl CLR, der dann aber sämtliche Variablen löscht. Mit dem Programm »CLEAR« (Listing 1) können Sie ein einzelnes Variablenfeld löschen. Somit ist es möglich, diese Variable neu zu dimensionieren. Mit SYS 49152 wird der CLR-Befehl erweitert. Folgt nun nach CLR ein Variablenname (zum Beispiel CLR A$), so wird diese Variable (falls vorhanden) aus dem Array-Bereich entfernt. Die Funktionsweise des Programms kann dem Assemblerlisting entnommen werden (Listing 2).

Wie verwaltet der Computer nun ein Variablenfeld? Im Bild rechts sehen Sie, wie die drei möglichen Variablentypen im Speicher abgelegt werden. In den ersten beiden Bytes steht der Variablenname. Besteht der Name aus nur einem Buchstaben, so wird für den zweiten Namen der Wert 0 eingesetzt. Die drei Variablentypen erkennt der Computer daran, das bei einer Stringvariablen im zweiten Byte das Bit 7 gesetzt ist. Zu dem ASCII-Wert des Variablennamens wird also der Wert 128 dazugezählt. Bei einer Integervariablen ist im ersten und zweiten Byte das Bit 7 gesetzt. In den nächsten beiden Bytes steht die Anzahl der Bytes, die das Feld benötigt. Danach folgt die Anzahl der Dimensionen und die Anzahl der einzelnen Felder. Zum Schluß finden wir noch die Variablen selbst. Wie im Bild zu sehen ist, belegt eine Stringvariable 3 Byte, eine Integervariable 2 Byte und eine Fließkommavariable 5 Byte. Man kann also Speicherplatz sparen, wenn nach Möglichkeit Integervariablen benutzt werden. Dies ist aber nur bei Variablenfeldern der Fall, weil bei nichtdimensionierten Variablen immer 7 Byte pro Variable belegt werden.

(H. Kunz/og)

```
programm : clear-dim                c000 c0de
---------------------------------------------
c000 : a9 0b 8d 08 03 a9 c0 8d  2f
c008 : 09 03 60 20 73 00 c9 9c  47
c010 : f0 0f 20 79 00 4c e7 a7  10
c018 : 4c 08 af 20 60 a6 4c ae  22
c020 : a7 a9 00 85 f7 85 f8 20  1c
c028 : 73 00 f0 ef 20 13 b1 90  58
c030 : e7 20 79 00 85 f7 20 73  05
c038 : 00 f0 2b c9 24 f0 1e c9  8a
c040 : 25 f0 0f 85 f8 20 73 00  b0
c048 : f0 1c c9 24 f0 0f c9 25  36
c050 : d0 f3 20 73 00 a9 80 05  ea
c058 : f7 85 f7 d0 03 20 73 00  29
c060 : a9 80 05 f8 85 f8 a5 2f  bf
c068 : a6 30 85 f9 86 fa a5 fa  94
c070 : c5 32 d0 06 a5 f9 c5 31  e7
c078 : f0 34 a0 00 b1 f9 c5 f7  9c
c080 : d0 07 c8 b1 f9 c5 f8 f0  d0
c088 : 06 20 c6 c0 4c 6e c0 a5  ee
c090 : f9 a6 fa 85 f7 86 f8 20  23
c098 : c6 c0 a5 fa c5 32 d0 11  da
c0a0 : a5 f9 c5 31 d0 0b a5 f7  c5
c0a8 : 85 31 a5 f8 85 32 4c ae  c7
c0b0 : a7 a0 00 b1 f9 91 f7 e6  b7
c0b8 : f7 d0 02 e6 f8 e6 f9 d0  c5
c0c0 : d9 e6 fa 4c 9a c0 a0 03  8d
c0c8 : b1 f9 aa 88 b1 f9 18 65  48
c0d0 : f9 90 02 e6 fa 85 f9 8a  48
c0d8 : 18 65 fa 85 fa 60 ff fc  bf
```

Listing 1. »CLEAR-DIM« bitte mit dem MSE eingeben



Aufbau der verschiedenen Array-Typen

```
10 "; ====================================================
20 ";
30 ";                      PROGRAMM:  CLEAR
40 ";
50 ";                      LOESCHEN EINES EINZELNEN
60 ";                      VARIABLENFELDES MIT :
70 ";                      CLR X
80 ";
90 "; ====================================================
100 " .BA 49152
105 "         LDA #$0B  ; VEKTOREN AUF
110 "         STA $0308 ;
115 "         LDA #$C0  ; NEUE ROUTINE
120 "         STA $0309 ;
125 "         RTS       ;
130 ";===============================================
135 "         JSR $0073 ; NAECHSTES ZEICHEN
140 "         CMP #$9C  ; IST ES CLR
145 "         BEQ JA    ;
150 "         JSR $0079 ; LETZTES ZEICHEN'HOLEN
155 "         JMP $A7E7 ; BEFEHL AUSFUEHREN
160 "ERROR    JMP $AF08 ; SYNTAX ERROR AUSGEBEN
165 "NCLR     JSR $A660 ; NORMALEN CLR BEFEHL AUSFUEHREN
170 "         JMP $A7AE ; ZUR INTERPRETERSCHLEIFE
175 "JA       LDA #$00  ;
180 "         STA $F7   ; VARIABLENNAME AUF
185 "         STA $F8   ; NULL SETZEN
190 "         JSR $0073 ; NAECHSTES ZEICHEN HOLEN
195 "         BEQ NCLR  ; NICHTS, DANN NORMALEN CLR AUSFUEHREN
200 "         JSR $B113 ; PRUEFT AUF BUCHSTABE
205 "         BCC ERROR ; KEIN BUCHSTABE, DANN FEHLER
210 "         JSR $0079 ; LETZTES ZEICHEN NACH ACCU
215 "         STA $F7   ; UND NACH $F7
220 "         JSR $0073 ; NAECHSTES ZEICHEN
225 "         BEQ X2    ; NICHTS, DANN NUMERISCHE VARIABLE
230 "         CMP #'$   ; STRINGVARIABLE ?
235 "         BEQ STRING
240 "         CMP #'%   ; INTEGERVARIABLE
245 "         BEQ INTEGER
250 "         STA $F8   ; ZEICHEN NACH $F8
255 "LLX      JSR $0073 ; NAECHSTES ZEICHEN
260 "         BEQ X2    ; NICHTS, DANN NUMERISCHE VARIABLE
265 "         CMP #'$   ; STRINGVARIABLE ?
270 "         BEQ STRING
275 "         CMP #'%   ; INTEGERVARIABLE ?
280 "         BNE LLX   ;
285 "INTEGER  JSR $0073 ; NAECHSTES ZEICHEN
290 "         LDA #$80  ; BIT 7 IM 1. BYTE
295 "         ORA $F7   ; DES VARIABLENNAMEN
300 "         STA $F7   ; SETZEN
305 "         BNE SPRUNG
310 "STRING   JSR $0073 ; NAECHSTES ZEICHEN
315 "SPRUNG   LDA #$80  ; BIT 7 IM 2. BYTE
320 "         ORA $F8   ; DES VARIABLENNAMEN
325 "         STA $F8   ; SETZEN
330 "X2       LDA $2F   ; ============================
335 "         LDX $30   ; ARRAYANFANG NACH           I
340 "         STA $F9   ; $F9/$FA SPEICHERN          I
345 "         STX $FA   ; ============================
350 "LOOP     LDA $FA   ; ============================
355 "         CMP $32   ; $F9/$FA MIT ARRAY ENDE     I
360 "         BNE NO1   ;                            I
365 "         LDA $F9   ; VERGLEICHEN                I
370 "         CMP $31   ; ============================
375 "         BEQ NOFOUND
380 "NO1      LDY #$00     ; ERSTES BYTE DES
385 "         LDA ($F9),Y ; VARIABLENNAMENS
390 "         CMP $F7      ; MIT GESUCHTER VARIABLE
395 "         BNE NO       ; VERGLEICHEN
400 "         INY          ;
405 "         LDA ($F9),Y ; ZWEITES BYTE
410 "         CMP $F8      ; VERGLEICHEN
415 "         BEQ FOUND    ;
420 "NO       JSR PLUS     ; NAECHSTE VARIABLE
425 "         JMP LOOP     ;
430 "FOUND    LDA $F9      ; VARIABLE GEFUNDEN
435 "         LDX $FA      ; ANFANG DER GESUCHTEN VARIABLE
440 "         STA $F7      ; NACH $F7/$F8 SPEICHERN
445 "         STX $F8
450 "         JSR PLUS     ; ANFANG DER NAECHSTEN VARIABLEN
455 "N2       LDA $FA      ; ==============================
460 "         CMP $32      ;                              I
465 "         BNE VERSCH   ; ENDE DES ARRAYBEREICHS ?     I
470 "         LDA $F9      ;                              I
475 "         CMP $31      ; ==============================
480 "         BNE VERSCH   ;
485 "         LDA $F7      ; ZEIGER ARRAY ENDE
490 "         STA $31      ;
495 "         LDA $F8      ; SETZEN
500 "         STA $32      ;
505 "NOFOUND  JMP $A7AE    ; ZUR INTERPRETERSCHLEIFE
510 "VERSCH   LDY #$00     ; ====================================
515 "         LDA ($F9),Y ;                                      I
520 "         STA ($F7),Y ; ANFANG DER NAECHSTEN VARIABLEN      I
525 "         INC $F7      ; NACH ANFANG DER GESUCHTEN VARIABLEN I
530 "         BNE N1       ; VERSCHIEBEN, BIS ARRAY ENDE         I
535 "         INC $F8      ; ERREICHT IST.                       I
540 "N1       INC $F9      ;                                     I
545 "         BNE N2       ;                                     I
550 "         INC $FA      ;                                     I
555 "         JMP N2       ; ====================================
560 "PLUS     LDY #$03     ; ==============================
565 "         LDA ($F9),Y ;                               I
570 "         TAX          ;                              I
575 "         DEY          ;                              I
580 "         LDA ($F9),Y ;                               I
585 "         CLC          ; ANFANG DER NAECHSTEN         I
590 "         ADC $F9      ;                              I
595 "         BCC N3       ; VARIABLEN ERRECHNEN          I
600 "         INC $FA      ;                              I
605 "N3       STA $F9      ; UND NACH $F9/$FA SPEICHERN   I
610 "         TXA          ;                              I
615 "         CLC          ;                              I
620 "         ADC $FA      ;                              I
625 "         STA $FA      ;                              I
630 "         RTS          ; ==============================
```

Listing 2. Assembler-Listing zu »CLEAR-DIM«

Tornado-Tape: so schnell wie der Blitz

»Tornado-Tape« hat es geschafft. Endlich ist die Datasette schneller als das Disketten-Laufwerk 1541.

Tornado-Tape« ist das schnellste zur Zeit erhältliche Programm zum Laden und Speichern von und auf Kassette. Beim Arbeiten mit diesem Programm ist unbedingt darauf zu achten, daß gutes Bandmaterial verwendet wird, denn wegen der extrem hohen Schreib- und Ladegeschwindigkeit müßten Sie bei billigem Bandmaterial mit Datenverlust rechnen. Außerdem muß das Datasettenkabel geerdet werden, um eventuelle Störungen von »außen« zu vermeiden. Die Bedienung des Programms »Tornado-Tape« ist denkbar einfach. Nach der Eingabe mit dem MSE wird das Programm mit SYS 49152 aktiviert. Anschließend stehen neben den schnellen Lade- und Speicher-Routinen auch die normalen zur Verfügung. Soll ein Programm im »Tornado-Tape«-Format geladen oder gespeichert werden, dann ist nur der Befehl SAVE, LOAD oder VERIFY einzugeben, also ohne jegliche Parameter. Auf die Angabe eines Filenamens wurde verzichtet, da sich Datasetten-Benutzer ohnehin zum jeweiligen Programm den Zählerstand merken müssen. Wird hinter den einzelnen Befehlen jedoch ein Filename mit angegeben, so erwartet Tornado-Tape ein Programm im langsamen Format.

Wird während des Arbeitens mit »Tornado-Tape« ein RESET ausgelöst oder die Tastenkombination RUN-STOP/RESTORE gedrückt, so muß das Programm mit dem Basic-Befehl SYS 49152 neu aktiviert werden.

Die Funktionstasten sind beim Arbeiten mit dem schnellsten Turbotape wie folgt belegt:

F1 = SAVE; F3 = LOAD; F5 = VERIFY.

Ansonsten bleibt nur, Ihnen mit Tornado-Tape viel Spaß und kurze Ladezeiten zu wünschen.

(John Bettels/ah)

```
programm : tornado-tape      c000 c530
---------------------------------------

c000 : ae 86 02 a0 00 84 9e 8c   3e
c008 : 20 d0 8c 21 d0 b9 9a c4   a6
c010 : 20 d2 ff c8 c0 47 d0 f5   28
c018 : 8e 86 02 a9 58 8d 30 03   58
c020 : 78 a9 3f 8d 14 03 a9 c0   70
c028 : 8d 15 03 58 a9 c0 8d 31   45
c030 : 03 a9 6c 8d 32 03 a9 c0   38
c038 : 8d 33 03 20 f8 c4 60 a4   a4
c040 : c5 c0 40 f0 20 a5 9e d0   df
c048 : 0c c0 04 f0 31 c0 05 f0   e3
c050 : 26 c0 06 f0 30 4c 31 ea   76
c058 : 85 93 a9 00 85 02 a5 b7   80
c060 : f0 3f 4c a7 f4 a9 00 85   9f
c068 : 9e 4c 31 ea a9 01 85 02   93
c070 : a5 b7 f0 2d 4c ed f5 a2   24
c078 : 4c a0 cf 4c 8d c0 a2 53   a2
c080 : a0 c1 4c 8d c0 a2 56 a0   81
c088 : c5 a9 01 85 93 8e 77 02   a2
c090 : 8c 78 02 a9 0d 8d 79 02   35
c098 : a9 03 85 c6 85 9e 4c 31   de
c0a0 : ea a5 02 f0 0f a5 2d c9   62
c0a8 : 03 d0 09 a5 2e c9 08 d0   fd
c0b0 : 03 68 68 60 20 a3 fd ad   80
c0b8 : 20 d0 85 a7 a9 fd 8d fd   53
c0c0 : 07 a2 32 8e 01 d4 a2 0c   d0
c0c8 : 8e 05 d4 85 f9 a9 07 85   d3
c0d0 : fa a9 00 8d 17 d4 8d 04   a7
c0d8 : d4 a9 0f 8d 18 d4 a9 37   34
c0e0 : 85 01 a5 02 d0 03 4c 8f   05
c0e8 : c2 a0 24 b9 59 c4 20 d2   1c
c0f0 : ff 88 10 f7 78 a5 01 29   41
c0f8 : 10 d0 fa a0 51 20 2f f1   fa
c100 : a9 00 85 fb 85 b5 a5 01   29
c108 : 29 1f 85 01 85 c0 a9 0e   63
c110 : 8d 20 d0 a9 00 8d 05 dc   51
c118 : 8d 11 d0 a8 a2 e6 ee 20   d4
c120 : d0 0a 88 d0 f9 ca d0 f6   59
c128 : a9 01 85 02 a5 01 29 f7   ea
c130 : 85 01 a9 85 8d 14 03 a9   2a
c138 : c1 8d 15 03 a9 7f 8d 0d   4c
c140 : dc a9 81 8d 0d dc a9 01   63
c148 : 85 fe 85 02 a5 2d 8d fe   e6
c150 : 07 18 69 03 85 2d a5 2e   d3
c158 : 8d ff 07 69 00 85 2e a9   0c
c160 : b1 8d 04 dc a9 01 85 d8   df
c168 : 58 ad 20 d0 29 0f c9 0e   07
c170 : f0 07 c9 05 f0 03 4c 69   22
c178 : c1 68 68 a9 11 8d 04 d4   f4
c180 : a6 d8 d0 fc 60 a5 01 49   30
c188 : 08 85 01 e6 b5 f0 17 a5   fa
c190 : b5 c9 ff f0 09 ad 0d dc   34
c198 : 68 a8 68 aa 68 40 a9 19   25
c1a0 : 8d 04 dc 4c 95 c1 a9 b3   65
c1a8 : 8d 14 03 a9 c1 8d 15 03   18
c1b0 : 4c 95 c1 a5 01 49 08 85   71
c1b8 : 01 49 07 f0 09 ad 0d dc   2a
c1c0 : 68 a8 68 aa 68 40 26 fd   09
c1c8 : b0 17 a9 19 8d 04 dc a9   51
c1d0 : 11 8d 0e dc ae 0d dc c6   1b
c1d8 : fe f0 1d 68 a8 68 aa 68   ec
c1e0 : 40 a9 7d 8d 04 dc a9 11   f6
c1e8 : 8d 0e dc ad 0d dc c6 fe   3a
c1f0 : f0 06 68 a8 68 aa 68 40   10
c1f8 : a0 00 b1 f9 a0 08 84 fe   9e
c200 : 85 fd a5 f9 18 69 01 85   08
c208 : f9 a5 fa 69 00 85 fa c5   63
c210 : 2e f0 70 68 a8 68 aa 68   29
c218 : 40 a5 2d 38 e9 03 85 2d   a4
c220 : a5 2e e9 00 85 2e a2 64   74
c228 : 88 d0 fd ca d0 fa a0 01   5a
c230 : 8c 0e dc 88 84 33 a9 3f   12
c238 : 8d 14 03 a9 c0 8d 15 03   98
c240 : 84 c6 84 d8 8c 17 d4 a9   8c
c248 : 7f 8d 0d dc a9 81 8d 0d   64
c250 : dc a9 40 8d 05 dc a5 01   92
c258 : 09 23 85 01 20 f8 c4 85   5c
c260 : c0 a9 1b 8d 11 d0 a5 2d   f6
c268 : 85 31 85 2f a5 2e 85 32   13
c270 : 85 30 a5 a7 8d 20 d0 a9   dc
c278 : 90 85 34 a9 00 8d 18 d4   84
c280 : 4c 98 c1 a5 f9 c5 2d f0   a2
c288 : 90 68 a8 68 aa 68 40 a0   b4
c290 : 1b b9 7e c4 20 d2 ff 88   6a
c298 : 10 f7 a5 01 29 10 d0 fa   7a
c2a0 : a5 01 29 df 85 01 85 c0   04
c2a8 : 20 d2 f5 a9 00 8d fe 07   5a
c2b0 : 8d ff 07 8d 20 d0 8d 11   91
c2b8 : d0 85 b5 78 a9 e9 8d 14   0f
c2c0 : 03 a9 c2 8d 15 03 a9 01   0c
c2c8 : 85 fd a9 7f 8d 0d dc a9   ae
c2d0 : 90 8d 0d dc a9 ff 8d 05   e1
c2d8 : dc a9 00 85 9c 8d 04 dc   3a
c2e0 : 85 b5 a9 02 85 02 4c 64   4d
c2e8 : c1 ad 04 dc 30 18 a5 b5   e2
c2f0 : c9 6d b0 17 a9 00 85 b5   9b
c2f8 : a9 11 8d 0e dc ad 0d dc   78
c300 : 68 a8 68 aa 68 40 e6 b5   bb
c308 : 4c f8 c2 a9 2a 8d 14 03   1b
c310 : a9 c3 8d 15 03 4c f8 c2   9d
c318 : a9 a6 8d 14 03 a9 c3 8d   a2
c320 : 15 03 a9 05 8d 20 d0 4c   78
c328 : f8 c2 a5 93 d0 ea a9 40   e9
c330 : 8d 14 03 a9 c3 8d 15 03   c0
c338 : a9 0e 8d 20 d0 4c f8 c2   29
c340 : ad 04 dc 30 13 a9 11 8d   0a
c348 : 0e dc ad 0d dc 18 26 fd   f4
c350 : b0 19 68 a8 68 aa 68 40   ba
c358 : a9 11 8d 0e dc ad 0d dc   d8
c360 : 38 26 fd b0 06 68 a8 68   58
c368 : aa 68 40 a0 00 a5 fd 91   b3
c370 : f9 a5 f9 18 69 01 85 f9   66
c378 : a5 fa 69 00 85 fa c8 84   51
c380 : fd cd ff 07 f0 06 68 a8   77
c388 : 68 aa 68 40 a5 f9 cd fe   c7
c390 : 07 f0 06 68 a8 68 aa 68   e7
c398 : 40 ad fe 07 85 2d ad ff   c8
c3a0 : 07 85 2e 4c 2e c2 ad 04   37
c3a8 : dc 30 13 a9 11 8d 0e dc   06
c3b0 : 18 26 fd b0 1c ad 0d dc   8e
c3b8 : 68 a8 68 aa 68 40 a9 11   35
c3c0 : 8d 0e dc 38 26 fd b0 09   ba
c3c8 : ad 0d dc 68 a8 68 aa 68   89
c3d0 : 40 a5 fd 45 9c 85 9c a5   bf
c3d8 : f9 18 69 01 85 f9 a5 fa   0d
c3e0 : 69 00 85 fa a0 01 84 fd   2a
c3e8 : ac 0d dc c5 2e f0 06 68   5e
c3f0 : a8 68 aa 68 40 a5 f9 c5   29
c3f8 : 2d f0 14 68 a8 68 aa 68   f9
c400 : 40 a0 0f b9 e1 c4 20 d2   f6
c408 : ff 88 10 f7 4c 2e c2 a5   db
c410 : 2d 8d fe 07 a5 2e 8d ff   a6
c418 : 07 a9 fd 85 a4 a9 07 85   e3
c420 : a5 a0 00 84 9b b1 a4 45   0a
c428 : 9b 85 9b a5 a4 18 69 01   d4
c430 : 85 a4 a5 a5 69 00 85 a5   1d
c438 : c5 2e f0 03 4c 25 c4 a5   fd
c440 : 2d c5 a4 d0 e0 a5 9b c5   c8
c448 : 9c d0 b6 a0 06 b9 f1 c4   8d
c450 : 20 d2 ff 88 10 f7 4c 2e   39
c458 : c2 0d 91 45 50 41 54 20   4e
c460 : 4f 44 41 4e 52 4f 54 20   1d
c468 : 4e 4f 20 59 41 4c 50 20   89
c470 : 26 20 44 52 4f 43 45 52   cb
c478 : 20 53 53 45 52 50 0d 91   be
c480 : 45 50 41 54 20 4f 44 41   d8
c488 : 4e 52 4f 54 20 4e 4f 20   50
c490 : 59 41 4c 50 20 53 53 45   1b
c498 : 52 50 9a 11 b9 b9 b9 b9   9f
c4a0 : 9b 54 4f 52 4e 41 44 4f   22
c4a8 : 20 54 41 50 45 9a b9 b9   d0
c4b0 : b9 b9 0d 1f b8 b8 b8 b8   13
c4b8 : 98 20 56 45 52 53 49 4f   22
c4c0 : 4e 31 2e 33 1f 20 b8 b8   e0
c4c8 : b8 b8 0d 20 20 20 20 20   e8
c4d0 : 31 39 38 35 20 36 34 27   25
c4d8 : 45 52 20 20 20 20 20 20   16
c4e0 : 0d 0d 91 21 52 4f 52 52   8a
c4e8 : 45 20 59 46 49 52 45 56   45
c4f0 : 0d 20 20 20 20 4b 4f 0d   cd
c4f8 : a9 3f 8d 18 d4 a9 f1 8d   25
c500 : 17 d4 a9 0c 8d 05 d4 a9   15
c508 : 10 8d 04 d4 a9 11 8d 04   dc
c510 : d4 ad 12 d0 85 a5 a0 00   61
c518 : ad 12 d0 49 55 8d 00 d4   97
c520 : ca d0 f5 98 45 a5 8d 01   9c
c528 : d4 8d 16 d4 88 d0 e9 60   5a
```

Listing zum Programm »Tornado-Tape«. Bitte beachten Sie die Eingabehinweise auf Seite 7.

Flottes Kopieren mit Express-Copy

Mit dem Programm Express-Copy kann man eine ganze Diskette in knapp drei Minuten kopieren. Dieses Programm ist also ideal für jeden, der ab und zu Sicherheitskopien von wichtigen Daten-Disketten anfertigen muß.

Wer sich schon immer ein schnelles Disketten-Kopierprogramm gewünscht hat, sich aber keines kaufen wollte, der findet hier genau das richtige Listing. Mit Express-Copy lassen sich ganze Disketten (und keine einzelnen Programme oder Dateien!) in knapp drei Minuten kopieren, Zeit für Diskettenwechsel nicht eingerechnet.

Noch ein wichtiger Hinweis: Ein einwandfreier Betrieb von »Express-Copy« kann nur bei Benutzung eines Original-ROMs in der Floppy garantiert werden. Besitzer von »Speed-Dos«, »Turbo Access« oder »Prologie Dos« müssen vielleicht auf Express-Copy verzichten.

Nachdem man Listing 1 mit dem MSE abgetippt und gespeichert hat, kann man das Programm mit »LOAD "EXPRESS-COPY",8« laden und mit »RUN« starten.

Sekundenbruchteile später erlischt der Bildschirm, der Laufwerksmotor läuft an und der Schreib/Lese-Kopf positioniert sich. Dies hat alles seine Richtigkeit, das Laufwerk also nicht abschalten!

Kurz darauf erwartet der Computer von Ihnen die Eingabe des zu kopierenden Track-Bereiches. Der Start- wie Endtrack darf dabei im Bereich von 01 bis 42 liegen. Wenn Ihnen diese Zahlen nichts sagen, geben Sie als Starttrack einfach »01« und als Endtrack einfach »35« ein. Dann werden alle normal beschriebenen Disketten automatisch richtig kopiert. Ihre Eingaben müssen Sie immer mit RETURN bestätigen. Falsche Eingaben können vor der Bestätigung mit der INST/DEL-Taste gelöscht werden.

Als nächstes werden Sie gefragt, ob die Zieldiskette formatiert werden soll. Bei schon beschriebenen Disketten, die nur mit einem neuen Inhalt überschrieben werden sollen, erübrigt sich das Formatieren (Achtung! Der alte Inhalt wird trotzdem dabei gelöscht). Da das Formatieren aber nur wenige Sekunden in Anspruch nimmt, sollte man es ruhig auch bei schon formatierten Disketten anwählen. Als Eingabe wird hier der Druck der Taste »Y« für Ja (Yes) oder N für Nein (No) angenommen.

Als letztes möchte der Computer noch von Ihnen wissen, ob er den Lesekopf (hier mit DC abgekürzt) während einzelner Leseversuche nachjustieren soll. Disketten, die mit leicht verstelltem Tonkopf beschrieben wurden, können durch ein Umkopieren auf eine andere Diskette mit gleichzeitiger Justierung »gerettet« werden. Dies muß allerdings nicht immer klappen. Beim Justieren wird bei Lesefehlern der Kopf leicht hin- und herbewegt, um den Disketten-Inhalt vielleicht noch richtig zu erwischen. Im Normalfall kann man aber auf eine Justage während des Lesens verzichten.

Nachdem alle Eingaben gemacht wurden, erlischt der Bildschirm für kurze Zeit. Kurz danach ertönt aus dem Lautsprecher des Fernsehers/Monitors ein sirenenartiges Geräusch. Gleichzeitig erscheint auf dem Bildschirm die Meldung »Read Track XX«. Dies ist die Aufforderung, die Diskette, von der kopiert werden soll, in das Laufwerk einzulegen. Danach drücken Sie bitte RETURN. Nun wird in zirka dreißig Sekunden ein Drittel der Diskette in den Speicher des C 64 gelesen. Während dieser Zeit ist der Bildschirm schwarz. Danach ertönt wieder die Sirene und es erscheint die Meldung »Write Track XX«. Nun müssen Sie die Diskette, auf die kopiert werden soll, einlegen und wieder RETURN drücken. Nach einer Weile erscheint dann wieder die Meldung »Read Track YY«, und Sie müssen wieder die erste Diskette einlegen, und so weiter...

Nach normalerweise dreimaligem Einlegen jeder Diskette erscheint die Meldung »End of Copy« und der Kopiervorgang ist beendet. Sollten Sie den Kopiervorgang vorzeitig abbrechen wollen, genügt ein Druck auf die Taste mit dem Pfeil nach oben, sobald der Computer »Read Track« oder »Write Track« anzeigt.

Ein paar Worte für die Profis: Express-Copy kopiert die Lesefehler 22 und 23 originalgetreu auf die Zieldiskette. Nicht kopiert werden die Fehler 20, 21 und 27. Der entsprechende Block auf der Diskette wird dann mit Nullen auf der Diskette gefüllt. Wenn die Tracks 36 bis 42 kopiert werden sollen, wird für diese die Lesegeschwindigkeit der Tracks 31 bis 35 verwendet. Damit können nicht alle Disketten, die auf diesen Tracks formatiert sind, kopiert werden! Manchmal werden diese Tracks nämlich in anderen Geschwindigkeiten formatiert, so zum Beispiel beim Format-System von Karsten Schramm aus dem Floppy-Kurs in der 64'er, Ausgabe 5/85, Seite 145.

Noch ein kleiner Hinweis: Dieses Kopierprogramm kann auch einige wenige kopiergeschützte Originalprogramme kopieren. Es sei kurz darauf hingewiesen, daß das Kopieren dieser Programme, außer für den privaten persönlichen Gebrauch, gesetzlich verboten ist.

(Daniel Gorrera/bs)

```
programm : express-copy    0801 1169
-------------------------------------

0801 : 0b 08 c1 07 9e 32 30 36  0a
0809 : 34 00 00 00 00 00 00 a9  91
0811 : 97 8d 00 dd a9 0b 8d 11  76
0819 : d0 a9 9d a2 0c 85 b2 86  3e
0821 : b3 a9 00 a2 03 85 ae 86  21
0829 : af a9 08 20 0c ed a9 6f  69
0831 : 20 b9 ed a9 4d 20 dd ed  07
0839 : a9 2d 20 dd ed a9 57 20  06
0841 : dd ed a0 00 a5 ae 20 dd  49
0849 : ed a5 af 20 dd ed a9 1e  29
0851 : 20 dd ed b1 b2 20 dd ed  91
0859 : c8 c0 1e d0 f6 20 fe ed  6b
0861 : 18 a5 b2 69 1e 85 b2 90  20
0869 : 03 e6 b3 18 a5 ae a6 af  99
0871 : 69 1e 85 ae 90 b3 e8 86  78
0879 : af e0 08 90 ac a9 08 20  25
0881 : 0c ed a9 6f 20 b9 ed a9  b7
0889 : 4d 20 dd ed a9 2d 20 dd  5c
0891 : ed a9 45 20 dd ed a9 86  a9
0899 : 20 dd ed a9 06 20 dd ed  0d
08a1 : 20 fe ed a9 00 8d 20 d0  7f
08a9 : 8d 21 d0 a9 1b 8d 11 d0  34
08b1 : a2 fd 86 8c e8 86 8e e8  54
08b9 : 86 3c e8 86 ae 20 6c 0b  1c
08c1 : 93 11 20 11 20 08 0e 9e  bf
08c9 : 20 c5 58 50 52 45 53 53  2f
08d1 : 2d c3 4f 50 59 0d 0d 20  30
08d9 : 20 20 28 43 29 20 27 38  1c
08e1 : 35 20 42 59 20 c4 2e c7  53
08e9 : 4f 52 52 45 52 41 0d 00  02
08f1 : 20 6c 0b 0d 0d 46 52 4f  97
08f9 : 4d 20 d4 52 41 43 4b 3a  a6
0901 : 00 20 11 0c 85 14 20 6c  2a
```

Listing zu »Express-Copy«. Bitte mit dem MSE eingeben. Beachten Sie die Eingabehinweise auf Seite 7.

```
0909 : 0b 0d 0d 20 20 54 4f 20   04
0911 : d4 52 41 43 4b 3a 00 20   8e
0919 : 11 0c 85 15 c5 14 b0 03   fa
0921 : 4c ac 08 20 6c 0b 0d 0d   37
0929 : 20 c6 4f 52 4d 41 54 20   3b
0931 : c4 49 53 4b 20 5b 59 2f   79
0939 : 4e 5d 3a 00 20 87 0c a2   78
0941 : 20 c9 59 f0 02 a2 2c 86   ad
0949 : fa 20 6c 0b 0d 0d 20 ca   1f
0951 : 55 53 54 49 43 45 20 c4   f7
0959 : c3 20 20 5b 59 2f 4e 5d   a3
0961 : 3a 00 20 87 0c a2 00 c9   fe
0969 : 59 f0 02 a2 04 86 fb a2   b9
0971 : 80 a9 0b 8d 11 d0 88 d0   96
0979 : fd ca d0 fa a5 14 20 90   0b
0981 : 0b a5 15 20 90 0b a5 fa   96
0989 : 20 90 0b a5 fb 20 90 0b   82
0991 : 20 c5 0a a9 0d 20 d2 ff   69
0999 : c6 14 20 6c 0b 0d 20 20   d9
09a1 : d2 45 41 44 20 d4 52 41   63
09a9 : 43 4b 3a 00 e6 14 a5 14   ee
09b1 : 85 fa 20 d7 0a 20 f4 0a   40
09b9 : a9 00 a2 10 85 8b 85 8d   f3
09c1 : 85 3b 86 af 20 c2 0b f0   a1
09c9 : 2d 08 a0 00 91 3b e6 3b   27
09d1 : 28 30 1a 20 c2 0b 91 8b   7e
09d9 : e6 8b 20 c2 0b 91 8d e6   26
09e1 : 8d 20 c2 0b 91 ae c8 d0   e4
09e9 : f8 e6 af d0 d7 29 1f aa   f3
09f1 : ca 86 15 4c 8e 0a 20 c2   0c
09f9 : 0b 10 07 29 3f 85 14 4c   fc
0a01 : c5 09 20 c5 0a 20 6c 0b   75
0a09 : 0d 20 d7 52 49 54 45 20   f3
0a11 : d4 52 41 43 4b 3a 00 a5   99
0a19 : fa 20 d7 0a 20 f4 0a a9   80
0a21 : 00 85 8b 85 3b 85 8d a9   e1
0a29 : 10 85 af a0 00 20 c2 0b   1e
0a31 : d0 25 b1 3b 08 20 90 0b   41
0a39 : e6 3b 28 30 f0 b1 8b 20   d8
0a41 : 90 0b e6 8b b1 8d 20 90   ab
0a49 : 0b e6 8d b1 ae 20 90 0b   a5
0a51 : c8 d0 f8 e6 af d0 d6 10   99
0a59 : 07 29 3f 85 14 4c 2e 0a   e6
0a61 : c9 03 d0 06 20 c5 0a 4c   92
0a69 : 9b 09 c9 04 d0 1c 20 c5   76
0a71 : 0a 20 6c 0b 0d 20 20 d7   0a
0a79 : 52 49 54 45 50 52 4f 54   ab
0a81 : 45 43 54 20 4f 4e 00 4c   81
0a89 : 1d 0a 4c f4 0b a9 36 85   3f
0a91 : 01 20 6c 0b 0d d2 45 41   1e
0a99 : 44 45 52 52 4f 52 20 d4   10
0aa1 : 52 41 43 4b 3a 00 a5 14   30
0aa9 : 20 d7 0a 20 6c 0b 20 d3   83
0ab1 : 45 43 54 4f 52 3a 00 a5   d9
0ab9 : 15 20 d7 0a a9 34 85 01   6a
0ac1 : 78 4c c5 09 a9 36 85 01   56
0ac9 : a9 1b 8d 11 d0 ad 00 dd   bc
0ad1 : 09 03 8d 00 dd 60 f8 aa   d9
0ad9 : f0 08 a9 00 18 69 01 ca   9e
0ae1 : d0 fb d8 48 4a 4a 4a 4a   a3
0ae9 : 20 ed 0a 68 29 0f 09 30   1f
0af1 : 4c d2 ff 58 a2 00 86 c6   83
0af9 : a9 0d 20 d2 ff 8e 04 d4   b9
0b01 : 8e 02 d4 8e 05 d4 a9 08   45
0b09 : 8d 03 d4 a9 f0 8d 06 d4   bf
0b11 : a9 0f 8d 18 d4 8e 17 d4   70
0b19 : a9 11 8d 04 d4 a9 05 85   e8
0b21 : b3 a5 c6 d0 1e c6 02 a5   de
0b29 : 02 4a 90 f5 a5 03 8d 00   dc
0b31 : d4 e6 03 d0 ec e6 b3 a6   75
0b39 : b3 8e 01 d4 e8 e0 1e b0   7e
0b41 : dc 90 de a9 00 8d 04 d4   78
0b49 : a9 0b 8d 11 d0 a0 20 ca   25
0b51 : d0 fd 88 d0 fa 78 ad 77   75
0b59 : 02 48 20 90 0b 68 c9 5e   71
0b61 : f0 01 60 68 68 20 c5 0a   aa
0b69 : 4c ac 08 68 85 b2 68 85   b5
0b71 : b3 a0 00 e6 b2 d0 02 e6   d9
0b79 : b3 b1 b2 f0 09 20 d2 ff   ac
0b81 : e6 b2 d0 f5 f0 f1 e6 b2   53
0b89 : d0 02 e6 b3 6c b2 00 85   f2
0b91 : 02 a9 35 85 01 a9 0b 8d   0a
0b99 : 00 dd ad 00 dd 10 fb a9   95
0ba1 : 03 8d 00 dd a2 04 a9 03   1e
0ba9 : 46 02 6a 46 02 6a 4a 4a   85
0bb1 : ea 8d 00 dd ca d0 ef a2   56
0bb9 : 01 ca d0 fd a9 34 85 01   67
0bc1 : 60 a9 35 85 01 a9 0b 8d   98
0bc9 : 00 dd ad 00 dd 10 fb a9   c5
0bd1 : 03 8d 00 dd a2 05 ca ea   aa
0bd9 : d0 fc a2 04 ad 00 dd 0a   b7
0be1 : 08 0a 26 02 28 26 02 ca   0a
0be9 : d0 f2 a9 34 85 01 a5 02   1e
0bf1 : 49 ff 60 20 c5 0a 20 6c   5c
0bf9 : 0b 0d 11 20 c3 4f 50 59   7e
0c01 : 20 52 45 41 44 59 00 85   de
0c09 : c6 a5 c6 f0 fc 4c ac 08   66
0c11 : 18 a0 0b a6 d6 20 f0 ff   43
0c19 : a9 20 20 d2 ff 20 d2 ff   81
0c21 : 18 a0 0b a6 d6 20 f0 ff   53
0c29 : a9 00 85 02 a9 fe 85 03   23
0c31 : a9 00 85 c6 a5 c6 f0 fc   63
0c39 : ad 77 02 c9 14 f0 d1 38   dc
0c41 : e9 30 90 ec c9 0a b0 e8   85
0c49 : 48 06 02 06 02 06 02 06   3a
0c51 : 02 05 02 85 02 68 09 30   ef
0c59 : 20 d2 ff e6 03 d0 d1 a9   10
0c61 : 00 85 c6 a5 c6 f0 fc ad   cd
0c69 : 77 02 c9 14 f0 a2 c9 0d   3c
0c71 : d0 ed a5 02 f0 9a 38 f8   98
0c79 : a2 00 e8 e9 01 d0 fb d8   cb
0c81 : 8a c9 2b b0 8b 60 a9 00   33
0c89 : 85 c6 a5 c6 f0 fc ad 77   50
0c91 : 02 c9 4e f0 04 c9 59 d0   bf
0c99 : ed 4c d2 ff a5 43 10 10   36
0ca1 : a9 11 85 43 a9 00 85 44   d6
0ca9 : ad 00 1c 29 9f 8d 00 1c   21
0cb1 : a5 00 29 06 f0 11 c9 02   24
0cb9 : d0 03 4c 32 03 c9 04 d0   94
0cc1 : 03 4c c8 05 4c 6c 04 4c   8e
0cc9 : 9e fd 09 09 0a 0b a5 44   e1
0cd1 : 4a 4a 4a 4a 4a aa bd 2e   6a
0cd9 : 03 85 0c a5 77 d0 02 a5   a8
0ce1 : 43 85 0a a0 05 84 09 a0   57
0ce9 : 0b 2c 00 1c 30 12 ca d0   ee
0cf1 : f8 88 d0 f5 a9 04 85 78   e2
0cf9 : a9 99 20 ba 07 4c 9e fd   18
0d01 : a9 5a 85 4b a9 52 85 24   2e
0d09 : 20 56 f5 50 fe b8 ad 01   4a
0d11 : 1c c5 24 f0 09 c6 4b d0   cd
0d19 : ef a9 0a 4c 69 f9 a2 00   da
0d21 : 50 fe b8 ad 01 1c 95 25   66
0d29 : e8 e0 07 90 f3 20 97 f4   de
0d31 : a5 16 45 17 45 18 45 19   72
0d39 : 45 1a f0 07 c6 09 d0 c0   22
0d41 : 4c 1e f4 a5 18 c5 06 f0   38
0d49 : 03 4c 0b f4 85 22 a5 16   00
0d51 : a6 17 85 12 86 13 a6 19   f4
0d59 : e8 e8 e4 43 90 02 a2 00   fa
0d61 : a9 02 85 31 4c 5f 04 ca   f8
0d69 : d0 15 4c 3b 04 86 07 86   dc
0d71 : 19 8a 45 18 45 17 45 16   72
0d79 : 85 1a 20 34 f9 a2 5a 20   f8
0d81 : 56 f5 50 fe b8 ad 01 1c   fb
0d89 : d9 24 00 d0 da c8 c0 08   95
0d91 : d0 f0 20 56 f5 50 fe b8   fb
0d99 : ad 01 1c 91 30 c8 d0 f5   79
0da1 : a0 ba 50 fe b8 ad 01 1c   c7
0da9 : 99 00 01 c8 d0 f4 20 e0   93
0db1 : f8 e6 19 a5 19 20 6a 05   5e
0db9 : ea a5 3a 20 6a 05 a5 38   de
0dc1 : 20 6a 05 b9 00 02 20 6a   f4
0dc9 : 05 c8 d0 f7 ad 00 1c 49   43
0dd1 : 08 8d 00 1c a9 ff 2c a9   c2
0dd9 : 90 a6 07 09 80 9d 50 01   d8
0de1 : 08 a0 00 8c fe 02 28 10   8c
0de9 : 07 c6 0a d0 03 4c 9e fd   f9
0df1 : 8a 18 65 0c c5 43 90 02   1f
0df9 : e5 43 aa bd 50 01 10 05   39
0e01 : e8 8a 4c 58 04 4c d1 03   3c
0e09 : a0 00 84 0a a2 00 a5 39   3f
0e11 : 99 00 02 c8 c8 a5 0a 99   59
0e19 : 00 02 c8 a5 06 99 00 02   32
0e21 : c8 a5 13 99 00 02 c8 a5   32
0e29 : 12 99 00 02 c8 a9 0f 99   91
0e31 : 00 02 c8 99 00 02 c8 b9   3e
0e39 : fa 01 59 fb 01 59 fc 01   5a
0e41 : 59 fd 01 99 f9 01 e6 0a   64
0e49 : a5 0a c5 43 90 c0 98 48   cf
0e51 : a9 02 85 31 20 30 fe 68   d3
0e59 : a8 88 b9 00 02 99 45 02   ba
0e61 : 88 d0 f7 ad 00 02 8d 45   d6
0e69 : 02 20 f5 fd a9 00 85 32   cd
0e71 : ad 0c 1c 29 1f 09 c0 8d   a9
0e79 : 0c 1c a9 ff 8d 03 1c a9   b2
0e81 : 55 8d 01 1c a2 03 20 24   6c
0e89 : fe a9 ff 8d 01 1c a2 05   93
0e91 : 50 fe b8 ca d0 fa a2 0a   6b
0e99 : a4 32 50 fe b8 b9 00 02   a8
0ea1 : 8d 01 1c c8 ca d0 f3 a2   17
0ea9 : 09 50 fe b8 a9 55 8d 01   2e
0eb1 : 1c ca d0 f5 a9 ff a2 05   54
0eb9 : 50 fe b8 8d 01 1c ca d0   26
0ec1 : f7 a2 bb 50 fe b8 a9 55   09
0ec9 : 8d 01 1c e8 d0 f5 a0 00   3a
0ed1 : 50 fe b8 a9 55 8d 01 1c   02
0ed9 : 88 d0 f5 a9 55 a2 08 50   a7
0ee1 : fe b8 8d 01 1c ca d0 f7   0a
0ee9 : a5 32 18 69 0a 85 32 c6   fe
0ef1 : 0a d0 96 50 fe b8 50 fe   08
0ef9 : b8 20 00 fe a9 00 8d fe   70
0f01 : 02 85 50 4c 9e fd 85 08   63
0f09 : 2c 00 18 10 fb a9 10 8d   a6
0f11 : 00 18 2c 00 18 30 fb a2   61
0f19 : 04 a9 00 06 08 2a 0a 06   b9
0f21 : 08 2a 0a 8d 00 18 ca d0   00
0f29 : f0 ea ea ea ea ea a9 0f   71
0f31 : 8d 00 18 60 2c 00 18 10   14
0f39 : fb a9 10 8d 00 18 2c 00   30
0f41 : 18 30 fb a2 04 ca d0 fd   9a
0f49 : 8e 00 18 a2 04 ad 00 18   10
0f51 : 4a 08 4a 4a 66 08 28 66   8f
0f59 : 08 ca d0 f1 a9 0f 8d 00   82
0f61 : 18 a5 08 60 a5 43 85 0a   f9
0f69 : a9 00 20 6a 05 ea 20 98   c1
0f71 : 05 10 03 4c 68 06 aa ca   c0
0f79 : 86 07 a9 02 85 31 20 98   c1
0f81 : 05 85 3a 20 98 05 85 47   32
0f89 : a0 00 20 98 05 99 00 02   66
0f91 : c8 d0 f7 20 8f f7 ad fe   31
0f99 : 02 85 7c a9 00 8d fe 02   1f
0fa1 : 85 50 20 10 f5 a2 09 50   92
0fa9 : fe b8 ca d0 fa a9 ff 8d   e8
0fb1 : 03 1c ad 0c 1c 29 1f 09   49
0fb9 : c0 8d 0c 1c a9 ff a2 05   f6
0fc1 : 8d 01 1c b8 50 fe b8 ca   62
0fc9 : d0 fa a0 bb b9 00 01 50   f6
0fd1 : fe b8 8d 01 1c c8 d0 f4   e4
0fd9 : a5 7c 8d fe 02 b1 30 50   0f
0fe1 : fe b8 8d 01 1c c8 d0 f5   f6
0fe9 : 50 fe 20 00 fe ad 00 1c   56
0ff1 : 49 08 8d 00 1c a9 00 8d   cc
0ff9 : fe 02 c6 0a f0 03 4c cc   dd
1001 : 05 4c 9e fd 48 a9 00 aa   bb
1009 : 9d 00 02 e8 d0 fa 85 3a   b3
1011 : a9 07 85 47 a9 02 85 31   ab
1019 : 68 29 7f aa ca 86 07 4c   e1
1021 : f7 05 78 a9 15 8d 07 1c   00
1029 : a9 2d 85 06 a9 c2 20 c3   44
1031 : 07 a2 00 ca d0 fd a9 01   88
1039 : 85 06 a9 e0 20 c3 07 20   c4
1041 : 98 05 85 06 20 98 05 85   64
1049 : 99 e6 99 20 98 05 8d 89   bb
1051 : 07 20 98 05 8d e1 06 a5   7a
1059 : 06 85 0d 20 98 05 c9 5e   ff
1061 : d0 03 4c 95 06 a9 00 20   67
1069 : 6a 05 a5 06 09 80 20 6a   6a
1071 : 05 20 b8 07 a9 00 85 77   35
1079 : 85 83 85 10 a9 00 85 78   c5
1081 : a9 e2 20 c3 07 a2 00 86   ae
1089 : 09 bd 50 01 c9 ff d0 08   95
1091 : e8 e4 43 d0 f4 4c 1b 07   02
1099 : a4 78 b9 db fe f0 0a a9   56
10a1 : 00 9d 50 01 e6 09 4c f4   76
10a9 : 06 86 07 e8 8a 09 80 20   04
10b1 : 6a 05 a6 07 4c f4 06 a5   f8
10b9 : 09 f0 17 85 77 a5 83 38   d4
10c1 : f9 db fe 85 83 b9 db fe   8c
10c9 : 58 20 76 d6 78 e6 78 4c   e3
10d1 : e4 06 a5 83 58 20 76 d6   a0
10d9 : 78 e6 06 a5 06 c5 99 f0   d1
10e1 : 0f c9 0c f0 0b c9 17 f0   33
10e9 : 07 c9 24 f0 03 4c c9 06   c2
10f1 : a9 00 20 6a 05 a9 01 20   d2
10f9 : 6a 05 ea 20 98 05 c9 5e   3a
1101 : d0 03 4c 95 06 a5 0d 85   e5
1109 : 06 ad 00 1c 29 10 d0 08   d0
1111 : a9 04 20 6a 05 4c 5e 07   4c
1119 : a5 06 09 80 20 6a 05 a9   d0
1121 : 05 85 0e a9 e6 20 c3 07   2e
1129 : a9 e4 20 c3 07 e6 06 a5   d0
1131 : 06 c5 99 f0 17 c9 0c f0   70
1139 : 0b c9 17 f0 07 c9 24 f0   3e
1141 : 03 4c 7c 07 a9 03 20 6a   72
1149 : 05 4c bb 06 a9 24 20 6a   35
1151 : 05 4c 95 06 a9 00 a2 14   f0
1159 : 9d 50 01 ca 10 fa 60 85   1d
1161 : 00 58 a5 00 30 fc 78 60   84
```

Listing zu »Express-Copy« (Schluß)

Menü-gesteuertes Laden

In Zukunft laden Sie von Ihren Disketten nur noch ein einziges Programm: das Menü. Danach wählen Sie das gewünschte Programm direkt an und lassen es laden.

Komfortables Laden von der Diskette. Ohne langes Suchen nach dem Lade-Programm. Keine Frage mehr, ob das Programm mit »,8« oder mit »,8,1« geladen wird. Das Menü übernimmt diese Kleinigkeiten für Sie. Dabei benötigt es nur etwa sechs bis sieben Blöcke auf der Diskette (je nach Länge des Directories) und dürfte damit auf fast jeder Diskette Platz haben. Das Programm, das dieses Menü für Sie erstellt, ist der »MENUE-MAKER« (siehe Listing).

Bedienungsanleitung:

Sie laden den Menü-Maker und starten ihn mit RUN. Dann legen Sie die Diskette ein, für die Sie ein Menü erstellen wollen (ein eventueller Schreibschutz ist zu entfernen), und wählen Menüpunkt 2.

Das Programm lädt nun das Directory und zeigt das erste Programm auf der Diskette an. Wenn das Programm später mit »,8« geladen werden muß, dann drücken Sie F 1, wenn Sie es mit »,8,1« laden müssen, dann drücken Sie F 3. Wenn es sich um ein Unterprogramm oder sonst ein Programm handelt, das für sich selber nicht lauffähig ist, oder von einem Lader nachgeladen wird, dann drücken Sie F 5 (das Programm wird nicht ins Menü übernommen).

Wenn Sie so alle Programme auf der Diskette durchgegangen sind, erscheint wieder das Hauptmenü. Wählen Sie nun Punkt 4 und das Disk-Menü wird zusammen mit der Datei auf Diskette gespeichert. Sollte Ihnen bei Punkt 2 ein Fehler unterlaufen sein, dann wählen Sie diesen nochmals an und gehen ihn wieder durch, bis alles richtig ist. Dann fahren Sie weiter, wie oben beschrieben.

Zusatzfunktionen: Wenn Sie Menüpunkt 1 anwählen, können Sie sich das Directory anschauen. Diese Funktion hat keinen Einfluß auf Punkt 2 und 4.

Durch Wählen von Punkt 3 erhalten Sie eine kurze Anleitung. Mit Punkt 5 können Sie das Programm beenden. Zur Sicherheit werden Sie noch einmal gefragt, ob Sie das Programm wirklich verlassen wollen. Wenn ja, wird ein Reset ausgeführt. Im anderen Fall sind Sie wieder im Hauptmenü. Punkt 1 und 3 verlassen Sie durch Drücken einer beliebigen Taste.

Programmbeschreibung:

Die wesentlichen Aspekte des Programms sind Menüpunkt 2 und 4.

Zu 2: Aus dem Directory werden die Programmnamen eingelesen und der indizierten Variablen NA$ (IN) zugeordnet. Dann wartet das Programm, bis man eine der drei möglichen F-Tasten gedrückt hat. Wenn man F 1 gedrückt hat, wird NU$ (IN) gleich 8 gesetzt, damit das Programm später weiß, daß es dieses Programm mit »,8« laden muß. Entsprechend wird nach Drücken von F3 NU$ (IN) auf 81 gesetzt. Wird F5 gedrückt, so wird NA$ (IN) wieder gelöscht und der Zähler IN um 1 zurückgesetzt. Diese Prozedur wird fortgesetzt, bis alle Programme im Directory abgearbeitet sind und das Programm wieder zum Menü springt.

Zu 4: Hier werden nun zuerst NA$ (IN) (das den Programmnamen enthält) und NU$ (IN) (das die Information enthält, ob mit »,8« oder mit »,8,1« geladen werden soll) als sequentielle Datei mit dem Namen »MSD« gespeichert. Ist dies geschehen, wird der Bildschirm gelöscht und folgendes darauf geschrieben (was man jedoch nicht sieht, da es in der Hintergrundfarbe geschrieben wird):

RUN 1600 (3 Zeilen Abstand)
POKE 43,PEEK (61)+1: POKE 44, PEEK (62)
(2 Zeilen Abstand)
SAVE "MENUE",8(4 Zeilen Abstand)
POKE 43,1: POKE 44,8 (2 Zeilen Abstand)
GOTO 1260

Dann wird der Tastaturpuffer mit RETURN gefüllt, und durch einen END-Befehl fährt das Programm im Direktmodus fort.

Auf dem Bildschirm stehen nun jedoch die oben genannten Befehle; der Tastaturpuffer ist mit RETURN gefüllt. Also führt der Computer diese Befehle aus.

Durch die ersten beiden Befehle wird das ab Zeile 1610 stehende Menü vom Rest des Programms getrennt (genaue Beschreibung siehe Sonderheft 1/85 unter Tips und Tricks: »AntiMERGE« von Rudolf Schmid-Fabian) und durch den SAVE-Befehl gespeichert.

Ist dies geschehen, so gelangt der Computer zu den beiden nächsten POKEs, die diese AntiMERGEroutine wieder aufheben. Danach wird durch GOTO 1260 wieder ins Menü des Hauptprogrammes eingestiegen.

Damit wäre der Zweck dieses einfachen Programms schon erreicht, nämlich möglichst leicht und schnell ein Menü zu erstellen. Man muß also nur noch das Menü laden und mit Run starten. Nachdem die Datei nachgeladen worden ist, kann man mit der Cursortaste und Return das zu ladende Programm wählen.

Das Menü lädt das Programm ebenfalls in einem »simulierten« Direktmodus nach. Es wird einfach ein LOAD- und ein RUN-Befehl auf den Bildschirm geschrieben, der Tastaturpuffer mit RETURN gefüllt und das Menü mit NEW gelöscht (wobei zugleich auch in den Direktmodus gesprungen wird). Übrigens kann das Menü (ab Zeile 1610) den eigenen Vorstellungen angepaßt werden. Man muß nur darauf achten, daß es nach dem STOP-Befehl in Zeile 1600 steht, da sonst die AntiMERGEroutine das Menü nicht sauber abtrennt.

(S. Brülisauer/og)

```
10 DIM NA$(144),NU$(144)                 <206>
20 PRINT CHR$(142):PRINT CHR$(8)         <154>
30 GOTO 1020                             <034>
40 REM *** ANLEITUNG ***                 <052>
50 PRINT"{CLR}";:PRINT                   <170>
60 PRINT SPC(14)"{BLUE,RVSON}ANLEITUNG   <123>
70 PRINT                                 <172>
80 PRINT"{WHITE}- DIESES PROGRAMM STELLT E
   IN MENU FUER                          <086>
90 PRINT"{2SPACE}IHRE DISKETTEN HER UND SP
   EICHERT ES                            <001>
100 PRINT"{2SPACE}AB. SIE MUESSEN ALSO NAC
    HHER NUR NOCH                        <088>
110 PRINT"{2SPACE}DAS MENU LADEN,DAS PROGR
    AMM DAS SIE                          <190>
120 PRINT"{2SPACE}WUENSCHEN AUSWAEHLEN UND
     SCHON WIRD                          <115>
130 PRINT"{2SPACE}ES IN DEN RECHNER GELADE
    N.                                   <055>
140 PRINT                                <242>
150 PRINT SPC(11)"{BLUE,RVSON}VORGEHENSWEI
    SE                                   <056>
160 PRINT                                <006>
170 PRINT"{WHITE}- WAEHLEN SIE ZUERST MENU
    PUNKT 2 UND                          <188>
```

»MENUE-MAKER« lädt Programme menügesteuert

```
180 PRINT"{2SPACE}BESTIMMEN SIE WELCHE PRO
    GRAMME INS                              <164>
190 PRINT"{2SPACE}MENU AUFGENOMMEN WERDEN
    SOLLEN.                                 <134>
200 PRINT"{2SPACE}GLEICHZEITIG LEGEN SIE A
    UCH FEST,OB                             <032>
210 PRINT"{2SPACE}ES MIT ,8 ODER MIT ,8,1
    (MASCHINEN-                             <092>
220 PRINT"{2SPACE}PROGRAMME) GELADEN WERDE
    N SOLL.                                 <016>
230 PRINT"- WENN SIE DIESE ARBEIT ERLEDIGT
     HABEN,                                 <117>
240 PRINT"{2SPACE}SIND SIE AUTOMATISCH WIE
    DER IM HAUPT-                           <070>
250 PRINT"{2SPACE}MENU UND KOENNEN NUN MEN
    UPUNKT 4 WAEH-";                        <053>
260 PRINT"{2SPACE}LEN UM DAS MENU ABZUSPEI
    CHERN ODER                              <171>
270 PRINT"{2SPACE}FALLS IHNEN BEI PUNKT 2
    EIN FEHLER                              <222>
280 PRINT"{2SPACE}UNTERLAUFEN IST,NOCHMALS
     VON VORNE                              <071>
290 PRINT"{2SPACE}BEGINNEN.{UP}"            <233>
300 GET T$:IF T$=""THEN 300                 <017>
310 RETURN                                  <114>
320 :                                       <042>
330 :                                       <052>
340 REM *** UEBERNEHMEN ***                 <055>
350 PRINT"{CLR}";:A$="":T$="":IN=0:X=0:XX=
    0                                       <037>
360 PRINT"{RVSON}F1 = ,8{RVOFF,SPACE,RVSON
    }F3 = ,8,1{RVOFF,SPACE,RVSON}F5 =_NICH
    T UEBERNEHMEN{RVSON,DOWN}"              <082>
370 CLOSE 2:OPEN 2,8,15                     <188>
380 OPEN 1,8,0,"$0"                         <020>
390 GET#1,A$,B$                             <097>
400 GET#1,A$,B$                             <107>
410 GET#1,A$,B$                             <117>
420 C=0                                     <075>
430 IF A$<>"" THEN C=ASC(A$)                <044>
440 IF B$<>"" THEN C=C+ASC(B$)*256          <121>
450 GET#1,B$:IF ST<>0 THEN 540              <138>
460 IF B$<>CHR$(34)THEN GOTO 450            <062>
470 GET#1,B$:IF B$<>CHR$(34)AND B$<>"<"THE
    N A$=A$+B$:GOTO 550                     <155>
480 IF B$<>CHR$(34)THEN GOTO 470            <083>
490 IF X=0 THEN X=1:A$="":GOTO 520          <014>
500 A$=RIGHT$(A$,LEN(A$)-1)                 <048>
510 A$=LEFT$(A$,LEN(A$)):PRINT A$;:GOSUB 5
    70                                      <119>
520 GET#1,B$:IF B$=CHR$(32) THEN 520        <026>
530 IF ST=0 THEN 400                        <172>
540 CLOSE 1:PRINT"{CLR}";:RETURN            <070>
550 IF LEN(A$)>20 THEN GOTO 540             <050>
560 GOTO 470                                <116>
570 IN=IN+1:NA$(IN)=A$                      <109>
580 GET T$:IF T$=""THEN 580                 <109>
590 IF T$="{F1}"THEN NU$(IN)="8":POKE 214,
    PEEK(214):POKE 211,20:SYS 58640:PRINT"
    {RVSON},8{RVOFF}"                       <004>
600 IF T$="{F1}"THEN RETURN                 <185>
610 IF T$="{F3}"THEN NU$(IN)="81":POKE 214
    ,PEEK(214):POKE 211,20:SYS 58640:PRINT
    "{RVSON},8,1{RVOFF}"                    <085>
620 IF T$="{F3}"THEN RETURN                 <078>
630 IF T$="{F5}"THEN NA$(IN)="":IN=IN-1:PO
    KE 214,PEEK(214):POKE 211,20:SYS 58640  <014>
640 IF T$="{F5}"THEN PRINT"{RVSON}NICHT UE
    BERNOMMEN{RVOFF}":RETURN                <163>
650 GOTO 580                                <230>
660 RETURN                                  <210>
670 :                                       <138>
680 REM *** DIRECTORY ***                   <137>
690 PRINT"{CLR}";                           <120>
700 CLOSE 2:OPEN 2,8,15:X=0:A$="":B$="":C=
    0:LE$=""                                <096>
710 OPEN 1,8,0,"$0"                         <096>
720 GET#1,A$,B$                             <173>
730 GET#1,A$,B$                             <183>
740 GET#1,A$,B$                             <193>
750 C=0                                     <151>
760 IF A$<>"" THEN C=ASC(A$)                <120>
770 IF B$<>"" THEN C=C+ASC(B$)*256          <199>
780 IF X=0 THEN TB=2:GOTO 800               <073>
790 TB=5                                    <080>
800 PRINT MID$(STR$(C),2);TAB(TB);          <135>
810 GET#1,B$:IF ST<>0 THEN 950              <252>
820 IF B$<>CHR$(34)THEN GOTO 810            <169>
830 IF X=0 THEN PRINT"{RVSON}";             <193>
840 PRINT CHR$(34);:PRINT CHR$(34);:PRINT"
    {DEL}";                                 <208>
850 GET#1,B$:IF B$<>CHR$(34)AND X=0 THEN P
    RINT"{RVSON}"B$;:GOTO 850:GOTO 870      <092>
860 IF B$<>CHR$(34)THEN PRINT B$;:GOTO 850  <142>
870 IF X=0 THEN PRINT"{RVSON}";             <233>
880 PRINT CHR$(34);:PRINT CHR$(34);:PRINT"
    {DEL}";                                 <248>
890 GET#1,B$:IF B$=CHR$(32) THEN 890        <022>
900 PRINT TAB(24);:C$=""                    <232>
910 C$=C$+B$:GET#1,B$:IF B$<>""  THEN 910   <164>
920 LE$=LEFT$(C$,6):IF X=0 THEN X=1:GOSUB
    980:PRINT"{4LEFT,RVSON}"LE$:GOTO 940    <028>
930 PRINT LE$                               <072>
940 IF ST=0 THEN 730                        <081>
950 PRINT"{LEFT}BLOCKS FREE":CLOSE 1        <212>
960 GET T$:IF T$=""THEN 960                 <235>
970 T$="":RETURN                            <228>
980 LE=LEN(C$):LE=6-LE                      <037>
990 FOR I=1 TO LE:ZUS$=ZUS$+" ":NEXT        <071>
1000 LE$=ZU$+LE$:RETURN                     <033>
1010 REM *** TITEL ***                      <110>
1020 POKE 53280,6 :POKE 53281,14:PRINT CHR
     $(142)CHR$(147);                       <020>
1030 PRINT                                  <116>
1040 PRINT"{BLACK,SPACE}QQQ{5SPACE}QQQ QQQ
     QQQQ QQQ{4SPACE}QQ QQ{4SPACE}QQ        <225>
1050 PRINT" QQQQ{3SPACE}QQQQ QQQQQQQ QQQQ{
     3SPACE}QQ QQ{4SPACE}QQ                 <164>
1060 PRINT" QQ QQ QQ QQ QQ"SPC(6)"QQ QQ{2S
     PACE}QQ QQ{4SPACE}QQ                   <109>
1070 PRINT" QQ{2SPACE}QQQ{2SPACE}QQ QQ"SPC
     (6)"QQ{2SPACE}QQ QQ QQ{4SPACE}QQ       <129>
1080 PRINT" QQ{3SPACE}Q{3SPACE}QQ QQ"SPC(6
     )"QQ{3SPACE}QQQQ QQ{4SPACE}QQ          <019>
1090 PRINT" QQ"SPC(7)"QQ QQQQ{4SPACE}QQ{4S
     PACE}QQQ QQ{4SPACE}QQ                  <037>
1100 PRINT" QQ"SPC(7)"QQ QQ"SPC(6)"QQ{5SPA
     CE}QQ QQ{4SPACE}QQ                     <252>
1110 PRINT" QQ"SPC(7)"QQ QQ"SPC(6)"QQ{5SPA
     CE}QQ QQ{4SPACE}QQ                     <006>
1120 PRINT" QQ"SPC(7)"QQ QQQQQQQ QQ{5SPACE
     }QQ QQQQQQQQ                           <236>
1130 PRINT" QQ"SPC(7)"QQ QQQQQQQ QQ{5SPACE
     }QQ{2SPACE}QQQQQQ                      <226>
1140 PRINT:PRINT                            <076>
1150 PRINT"{2SPACE}WW{5SPACE}WW{2SPACE}WWW
     W{2SPACE}WW{2SPACE}W WWWWW WWWW        <238>
1160 PRINT"{2SPACE}WWW{3SPACE}WWW WW{2SPAC
     E}WW WW W{2SPACE}WW{4SPACE}WW{2SPACE}
     W                                      <096>
1170 PRINT"{2SPACE}WW W W WW WW{2SPACE}WW
     WWW{3SPACE}WW{4SPACE}WW{2SPACE}W       <106>
1180 PRINT"{2SPACE}WW{2SPACE}W{2SPACE}WW W
     WWWWW WWW{3SPACE}WWWW{2SPACE}WWWW      <113>
1190 PRINT"{2SPACE}WW{5SPACE}WW WW{2SPACE}
     WW WW W{2SPACE}WW{4SPACE}WW W          <025>
1200 PRINT"{2SPACE}WW{5SPACE}WW WW{2SPACE}
     WW WW{2SPACE}W WWWWW WW{2SPACE}W       <101>
1210 PRINT                                  <040>
1220 PRINT"{4SPACE}"SPC(12)"SIMON (TEL 071
     /85 46 22)";                           <137>
1230 PRINT"{5SPACE}WRITTEN BY"SPC(6)"&      <083>
1240 PRINT"{4SPACE}"SPC(12)"EDY{3SPACE}(TE
     L 071/85 50 46)";                      <218>
1250 GET A$:IF A$=""THEN 1250               <157>
1260 PRINT"{CLR}"                           <232>
1270 PRINT,"{2DOWN,WHITE,2SPACE}IHRE WAHL
     :"                                     <110>
1280 PRINT,"{2DOWN,2SPACE}1{2SPACE}DIRECTO
     RY EINLESEN"                           <226>
1290 PRINT,"{2DOWN,2SPACE}2{2SPACE}UEBERNE
     HMEN"                                  <018>
1300 PRINT,"{2DOWN,2SPACE}3{2SPACE}INFO"    <160>
1310 PRINT,"{2DOWN,2SPACE}4{2SPACE}MENU SP
     EICHERN"                               <033>
1320 PRINT,"{2DOWN,2SPACE}5{2SPACE}ENDE"    <089>
1330 GET A$ :IF A$=""THEN 1330              <046>
1340 IF A$="1"THEN GOSUB 690:GOTO 1260      <103>
1350 IF A$="2"THEN GOSUB 350:GOTO 1260      <154>
1360 IF A$="3"THEN GOSUB 50:GOTO 1260       <061>
1370 IF A$="4"THEN GOTO 1460                <011>
1380 IF A$="5"THEN 1410                     <079>
```

```
1390 GOTO 1330                                <204>
1400 REM *** ENDE ***                         <217>
1410 POKE 214,23:POKE 211,11:SYS 58640:PRI
     NT"{RVSON,BLUE}SIND SIE SICHER ?{UP}"   <087>
1420 GET T$:IF T$="N"THEN 1260                <231>
1430 IF T$="J"THEN SYS 64738                  <098>
1440 GOTO 1420                                <238>
1450 REM *** DATEI SPEICHERN ***              <022>
1460 CLOSE 15:OPEN 15,8,15:PRINT#15,"S0:MS
     D":CLOSE 15:CLOSE 2:OPEN 2,8,2,"MSD,S
     ,W"                                      <223>
1470 Z=0                                      <201>
1480 Z=Z+1:IF Z=145 THEN CLOSE 2:GOTO 1530   <001>
1490 IF NA$(Z)=""THEN CLOSE 2:GOTO 1530       <004>
1500 PRINT#2,NA$(Z);NU$(Z)                    <118>
1510 GOTO 1480                                <245>
1520 REM *** MENU SPEICHERN ***               <247>
1530 CLOSE 15:OPEN 15,8,15:PRINT#15,"S0:ME
     NU":CLOSE 15:PRINT"{CLR,LIG.BLUE}";:P
     RINT"RUN1600"                            <148>
1540 PRINT"{3DOWN}POKE43,PEEK(61)+1:POKE44
     ,PEEK(62)"                               <054>
1550 PRINT"{2DOWN}SAVE"CHR$(34)"MENU"CHR$(
     34)",8"                                  <233>
1560 PRINT"{4DOWN}POKE43,1:POKE44,8"          <021>
1570 PRINT"{2DOWN}GOTO1260"                   <136>
1580 POKE 631,19:POKE 632,13:POKE 633,13:P
     OKE 634,13:POKE 635,13:POKE 636,13       <214>
1590 POKE 637,13:POKE 198,7:END               <022>
1600 STOP                                     <142>
1610 POKE 53280,14:POKE 53281,6               <042>
1620 PRINT"{CLR,9DOWN,WHITE,2DOWN,11SPACE}
     ICH LADE DIE DATEI":Z=0:DIM NA$(144),
     NU$(144)                                 <174>
1630 CLOSE 2:OPEN 2,8,2,"MSD,S,R"             <101>
1640 Z=Z+1                                    <184>
1650 INPUT#2,NAA$                             <192>
1660 IF RIGHT$(NAA$,1)="8"THEN NA$(Z)=LEFT
     $(NAA$,LEN(NAA$)-1):NU$(Z)=",8"          <068>
1670 IF RIGHT$(NAA$,2)="81"THEN NA$(Z)=LEF
     T$(NAA$,LEN(NAA$)-2):NU$(Z)=",8,1        <134>
1680 IF ST=64 THEN CLOSE 2:GOTO 1700          <250>
1690 GOTO 1640                                <074>
1700 PRINT"{CLR}";:Z=0                        <014>
1710 PRINT"{2DOWN,14SPACE}W{RVSON}MENU-MAK
     ER{RVOFF}H"                              <059>
1720 PRINT"{DOWN,15SPACE}BY SIMON & EDY"      <014>
1730 PRINT TAB(5)"{2DOWN}CURSOR UP{4SPACE}
     NAECHSTES ELEMENT"                       <120>
1740 PRINT TAB(5)"{DOWN}CURSOR DOWN{2SPACE
     }VORHERIGES ELEMENT"                     <180>
1750 PRINT TAB(5)"{DOWN}RETURN{7SPACE}WAEH
     LEN"                                     <127>
1760 PRINT"{HOME,14DOWN,4SPACE}U**********
     ******I"                                 <129>
1770 PRINT"{4SPACE}-{16SPACE}-"               <141>
1780 PRINT"{4SPACE}J****************K"        <032>
1790 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T NA$(1):Z=1                             <182>
1800 GET T$:IF T$=""THEN 1800                 <235>
1810 IF T$="{UP}"THEN Z=Z+1:X=1:GOSUB 1890
     :GOTO 1800                               <140>
1820 IF T$="{DOWN}"THEN Z=Z-1:X=2:GOSUB 18
     90:GOTO 1800                             <122>
1830 IF T$=CHR$(13)THEN GOTO 1850             <175>
1840 GOTO 1800                                <130>
1850 PRINT"{CLR,BLUE}";:PRINT"LOAD"CHR$(34
     )NA$(Z)CHR$(34)NU$(Z)                    <176>
1860 PRINT"{4DOWN}POKE646,14:RUN"             <082>
1870 POKE 631,19:POKE 632,13:POKE 633,13:P
     OKE 198,3:NEW                            <124>
1880 REM *** UNTERPROG ***                    <115>
1890 IF Z=0 THEN Z=1:RETURN                   <039>
1900 IF NA$(Z)=""AND X=2 THEN Z=Z+1:RETURN    <118>
1910 IF Z=145 THEN Z=144:RETURN               <200>
1920 IF NA$(Z)=""AND X=1 THEN Z=Z-1:RETURN    <025>
1930 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T"{16SPACE}"                             <188>
1940 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T NA$(Z):RETURN                          <126>
```

© 64'er

»MENUE-MAKER« (Schluß). Beachten Sie bitte die Eingabehinweise auf Seite 6

Filemanager schafft Übersicht

Irgendwann steht wohl jeder C 64-Besitzer verzweifelt vor seiner Diskettensammlung und sucht ein bestimmtes Programm. Ohne eine entsprechende Liste kann das jedoch zu einer aufwendigen Suche führen. »Filemanager« schafft hier wirkungsvoll Abhilfe.

Filemanager sortiert die Einträge nach drei Kriterien. Dadurch entfällt ständiges Umsortieren. »Filemanager« (Listing) läuft nicht mit Hypra-Load zusammen.

Zum Programm

Fangen wir mit der Speicheraufteilung an. »Filemanager« belegt den Speicherplatz von $CO00 - $CFD9. Die Filenamen werden zwischen einer Variablen (Ugrenze) und $C000 abgelegt. Außerdem existieren noch drei Zeigerstapel mit den Adressen $CFD2, $DFEC, $EFF6. Diese Zeigerstapel bestehen auf 16-Bit Zeigern, die auf den Anfang der einzelnen Filenamen zeigen. Beim Sortieren werden nur diese Zeigerstapel sortiert. Die Filenamen selber bleiben an ihrer Stelle im Speicher. Ein Filename ist folgendermaßen aufgebaut.

1. Byte:

Bits 0–3: Länge–1 des eigentlichen Filenamens.
Bits 4–5: Übertrag der Blocklänge der Files.
Bits 6–7: Filetyp 00 = Prog, 01 = Seq., 10 = Usr, 11 = Rel.

2. Byte:

Blocklänge des Files.
Bytes 3–4: ID des Files
Byte 5: Länge gesamtes File-Element: Name des Files.

Belegte Speicherbereiche:

($CFC7) - $C000 = Bereich Speicherplatz für Filenamen
$C000 - $CE8F = Maschinenprogramm
$CE8F - $CFDB = Bereich Datentabellen und Variablen
$CFDC -($CFD3) = Bereich Zeigerstapel für Blook
$CFE6 -($CFD5) = Bereich Zeigerstapel für Ilook
$CFF0 -($CFD7) = Bereich Zeigerstapel für Alook

Die Liste, die auf Diskette abgespeichert wird, hat folgendes Format:

Am Anfang stehen die Anzahl Byte, die für die gesamten Filenamen ohne Zeiger benötigt werden, danach kommt die Anzahl der Filenamen selber. Nach diesen 4 Byte folgt dieses periodische Format I-mal:

2 Byte: I.ter Zeiger aus obersten Zeigerstapel-Ugrenze.
2 Byte: I.ter Zeiger aus mittleren Zeigerstapel-Ugrenze.
2 Byte: I.ter Zeiger aus unteren Zeigerstapel-Ugrenze.
5–20 Byte: I.ter Filename der ungeordneten Liste.

Beim Laden wird eine noch im Speicher befindliche Liste nicht gelöscht, das heißt: es wird immer dazugeladen (Merge). Da auch die Zeiger mit gespeichert werden, ist eine Programmliste, die vor dem Speichern geordnet war, auch sofort nach dem Laden nach Blocklänge, alphabetischer Reihenfolge sowie nach der ID sortiert.

Befehle von »Filemanager«:
Alle Befehle bis auf »@« werden zu Token umgewandelt und können somit abgekürzt werden.
»@«
»@« alleine liest den Fehlerkanal aus. Falls ein String folgt, wird dieser, falls er nicht mit »$« beginnt, über den Befehlkanal zur Floppy gesendet.
Beispiel: @"$t*=seq" listet alle sequentiellen Files, die mit »t« beginnen.

DMERGE
Dmerge benötigt keine weiteren Parameter. Das Inhaltsverzeichnis der Diskette wird in der Liste aufgenommen.

HELP
Keine Parameter. »Help« listet alle neuen Schlüsselwörter bis auf »@«.

SMERGE
Wie »Dmerge«, außer daß nur die Filenamen in der Liste aufgenommen werden, die sich noch nicht in ihr befinden. Dabei sind alle Vergleichkriterien relevant. Da die Liste bei jeden Filenamen neu durchsucht werden muß, ist »Smerge« besonders bei längerer Liste erheblich langsamer als »Dmerge«. »Smerge« eignet sich dazu, neue Programmnamen in der Liste aufzunehmen, wenn sich schon einige Programmnamen der eingelegten Diskette in der Liste befinden.

SLOOK
Listet das gesamte Inhaltsverzeichnis der Diskette. Filenamen, die sich schon in der Liste befinden, werden revers gelistet. Beim Suchen in der Liste wird die ID nicht berücksichtigt, da das gleiche Programm schon auf einer anderen Diskette mit anderer ID sein kann.

KILL
»Kill« löscht die gesamte Liste.

QUIT
Quit schaltet »Filemanager« aus. »Filemanager« kann mit »SYS 50000« reaktiviert werden, ohne eine eventuell im Speicher befindliche Liste zu löschen.

MEM
Mem gibt die Anzahl der noch freien Filenamen-Plätze, begrenzt durch die Anzahl Zeiger im Zeigerstapel, die Anzahl der in der Liste befindlichen Namen, den noch freien Speicherplatz für die Filenamen selber und den schon verbrauchten Speicherplatz an. Dann wird »Membot« und der Zeiger auf Anfang Filenamenspeicher ausgegeben.

MEMBOT Numerischer Ausdruck
Setzen des Zeigers auf Anfang Filenamenspeicher auf einen neuen Wert, ohne die Liste zu zerstören. Membot wird dazu genutzt, den Speicherplatz entsprechend den Anforderungen zu bemessen. »Out of Memory Error« wird ausgegeben, wenn:
– der Zeiger auf einen niedrigeren Wert gesetzt werden soll, als in 55/56 steht (Obergrenze Speicherplatz für Basic Interpreter).
– dadurch der Zeiger auf das erste nicht genutzte Byte der Liste (Spointer) $C000 =Ogrenze überschreitet.

Disk Numerischer Ausdruck
»Disk« setzt die Nummer des aktuellen IEC-Peripherie-Gerätes. Nach dem Laden mit Basic-Lader wird automatisch Disk PEEK (186) - durchgeführt. Da es bei falscher Eingabe nicht zum Programmabsturz kommen kann, wird auf eine Überprüfung des Wertes verzichtet.

MERGE String
Laden einer auf Diskette gespeicherten Liste in den Speicher.

PUT String
Speichert die Liste auf Diskette.

SORT
Sortiert die Liste alphabetisch, nach ID und nach Blocklänge.

WRITE Numerischer Ausdruck String Filetyp String
Liest die Blocklänge, den Namen, den Typ und die ID eines Programms in den Speicher ein.
Dabei ist zu beachten, daß keine Verknüpfungen erlaubt sind wie zum Beispiel für String: "abc"+"cde". In diesem Fall muß man einer Stringvariablen diesen Wert zuweisen: A$="abc"+"cde": Write12A$P "ID".
Beim Filetyp genügt es, den ersten Buchstaben anzugeben. In diesem Fall müssen aber zwei beliebige Zeichen folgen. »Illegal Name Error« wird bei nicht eindeutigen Angaben sowie bei Verwendung des Jokers ausgegeben.

SCRATCH Parameter, wie Write
Löscht alle Files, die der Eingabemaske entsprechen, aus der Liste. Verwendung des Jokers »*« ist erlaubt für alle Parameter. Statt eines Jokers kann für die ID ein Leerstring eingegeben werden, oder man kann die ID auch ganz weglassen. Beispiel: Es sollen alle Filenamen aus der Liste entfernt werden, die sich auf der Diskette mit der ID "AD" befinden und mit "File" beginnen:
Scratch* "File*" * "AD"
Bei Verwendung des Jokers beim Filetyp müssen nicht, siehe letztes Beispiel, dem Joker zwei Zeichen folgen. Die Anzahl der gelöschten Namen wird ausgegeben. Da für gelöschten Namen die Blockverschiebe-Routine und die Zeigerkorrektur-Routine, die wiederum dreimal die Verschiebe-Routine aufruft, benötigt wird, braucht »Scratch« meist sehr viel Zeit.

BLOOK Numerischer Ausdruck – Numerischer Ausdruck
Listet alle Namen, die der Eingabemaske entsprechen. Verkettung numerischer Ausdrücke ist nicht gestattet. Verwendung des Jokers ist erlaubt. Sonst Syntax, wie beim List-Befehl.
Beispiel: alle Namen listen
Blook oder Blook*

ALOOK String – String
Funktion und Syntax wie Blook

ILOOK
siehe Alook

Bei sortierter Liste werden bei »Blook« nach Blocklänge, bei »Alook« nach alphabetischer Reihenfolge, bei »Ilook« nach ID die Filenamen geordnet ausgegeben.

»Filemanager« wird mit dem MSE eingegeben. Das fertige Programm sollte vor dem ersten Start gespeichert und wieder geladen werden, da die Blockverschiebe-Routine des Basic-Laders den Zeiger auf Variablenanfang ($2d/$2) benutzt.

(Christoph Deeken/gk)

Die wichtigsten Unterprogramme

ACOMPARE $C053 - $C09F
Zeiger1 und Zeiger3 zeigen auf den Anfang je eines Elementes. Diese Elemente werden auf die alphabetische Reihenfolge hin verglichen. Das Ergebnis steht in C- und Z-Flag.

ACTIVATE $C350 - $C366
Aktivieren des File-Managers.

BASOUT $C2DA - $C2ED
Ausgabe eines Zeichens mit Verwaltung des Spaltenzählers Cursor.

BCOMPARE $C09f - $C0B7
Wie »Acompare«, nur Blocklänge als Kriterium

BEFEHLCH $C3F2 - $C40C
Öffnen des Befehls-Kanals der Floppy

CHKFILMEN $C4AA - $C4D3
Wie »Chknammem«, prüft auf Platz für Zeiger3 Elemente und (Zeiger1) Bytes.

CHKNAMMEM $C499 - $C4D3
Prüft auf Platz für Element in Akku1. Falls kein Platz vorhanden, dann »Out of memory error«, sonst Rückkehr in Hauptprogramm.

COMPARE $C0D6 - $C115
Wie »Acompare«, das Vergleichskriterium wird in die Bitmaske übergeben. Bit 3-0 sind Direktbits, Bit 7-4 sind Vorrangsbits. Die Bits werden von links nach rechts geprüft. Wird ein gesetztes Vorrangsbit gefunden, dann wird geprüft, ob das entsprechende Direktbit gesetzt ist. Falls das entsprechende Direktbit nicht gesetzt ist, wird das Vorrangsbit gelöscht, ansonsten wird die entsprechende Vergleichsroutine aufgerufen. Ist das Ergebnis Z=0, dann ist die Routine beendet. Sonst werden die entsprechenden Bits in Bitmaske gelöscht und der Vergleich beginnt von vorne, bis alle Bits gelöscht sind oder als Ergebnis bei einem Vergleich Z=0 herausgekommen ist.

CRETURN $C2D2 - $C2ED
Ausgabe von Carriage Return.

DIROPEN $C3C8 - $C3F2
Öffnen des Directories. Löschen des Status.

DOPEN $C3CE - $C3F2
Öffnen eines Lesefiles. Parameter für Filenamen in den Registern.

ERRORD $C420 - $C430
Auslesen des Fehlerkanals

ERROROP $C6B0 - $C6C0
Öffnen des Fehlerkanals der Floppy

FIRSTNAM $C566 - $C5E1
Öffnet Directory und holt erstes Element nach Akku1.

FLOPPY $C68F - $C6A8
Gibt String, dessen Adressen in $22 und dessen Länge in Speicher steht, auf den Befehlskanal der Floppy aus.

FLOPPYD $C692 - $C6A8
Wie Floppy, nur ohne Öffnen des Befehlskanals.

FNADRESS $C018 - $C041
Stackbot zeigt auf den Anfang eines Zeigerstapels. Index ist relativer Zeiger in diesem Stapel. Zuerst wird der Inhalt der Adresse »Stackbot +2index« nach Zeiger2 kopiert. Dann wird der Inhalt von Zeiger2 nach Zeiger1 gebracht.

GETAKKU $C623 - $C63C
Holt das Element, auf das Zeiger1 zeigt, nach Akku1.

GETLONGI $C041 - $C04A
Aus den ersten Byte von Akku1 werden die Bits, in denen die Namenslänge-1 steht, herausgefiltert (and #%00001111) und in Namlog1 gespeichert.
Chklong1 $C044 - $C04A
Wie Getlong1, nur daß der Inhalt des A-Registers übergeben wird.

GETLONG2 $C04A - $C053
Wie Getlong1, nur mit Akku2 und Namlog2.
Chklong2 $C04D - $C053
Wie Getlong2, aber der Akkumulator wird übergeben.

HOCHKOM $C2D8 - $C2ED
Ausgabe eines Hochkommas.

ICOMPARE $C0B7 - $C0C5
Wie Acompare, ID als Kriterium.

IECINI $C40C - $C416
Wenn Status =0, dann Fehlerkanal auslesen und nächsten Befehl interpretieren, sonst Byte vom IEC-bus holen.

INTERRUPT $C345 - $C34C
Einschalten des ROMs und Sprung zum Basic-Kaltstart. (Wird benutzt, um bei abgeschalteten ROM ein Abstürzen bei Drücken der Restore-Taste zu vermeiden).

KEYSTR $C0D8 - $C1F4
Das X-te Wort aus Newtab wird ausgegeben.

LEERCHAR $C2D5 - $C2ED
Ausgabe eines Spaces.

MEMFREE $C479 - $C489
Freier Speicherplatz nach Zeiger2.

MEMFULL $C465 - $C479
Voller Speicherplatz nach Hzeiger.

NAMFREE $C489 - $C499
Anzahl freier Elemente nach Zeiger4.

NEWLIST $C1F4 - $C212
List-Routine für die neuen Token.

NEXTIND $C60B - $C623
Index - Index+1. Wenn Index = Counter, dann Z = 1.

NEXTNAM $C57F - $C5E1
Holt nächstes Element des Directories nach Akku1.

PRINTHEX $C212 - $C246
Die 16-Bitzahl A/X wird ausgegeben und bis zur 6. Ziffer mit Spaces aufgefüllt.

PRINTNAM $C246 - $C2AB
Das Element in Akku1 wird ausgegeben. Prüfung auf Stop-Taste. Warteschleife auf Tastendruck.

ROMAUS $C000 - $C00C
Umschalten auf RAM

ROMEIN $C00C - $C018
Umschalten auf ROM

SETGRENZ $C5ED - $C602
Füllt Akku2 mit $00 und Akku3 mit $ff.

SETINDEX $C5EF - $C5F8
Index =-1.

STORREAKKU $C4D3 - $C566
Legt Element in Akku1 in Liste ab.
TAB $C2AB - $C2CB
Ausgabe von Spaces bis zur nächsten Zehnertabulatur.
TCOMPARE $C0C5 - $C0D6
Filetyp als Kriterium, sonst wie »Acompare«.
TOKENNEU $C115 - $C1D8
Einbindung der neuen Token.
TRANSFER $C2ED - $C345 und $C366 - $C3AE
Der Bereich Zeiger2 bis Zeiger4 wird um Index Byte verschoben. Dabei wird Index als vorzeichenbehaftete Zahl benutzt.
DECTXTPT $C6A8 - $C6B0
Decrementiert den Programmzähler ($7a).
INNAM $C6C0 - $C702
Holt Filename vom Bildschirm nach Akku1.
INGRENZ $C702 - $C73F
Holt Grenzwerte vom Bildschirm nach Akku2 und Akku3.
A1NACHA3 $C730 - $C73C
Kopiert Akku1 nach Akku3.
INGRENZ6 $C73C - $C73F
Sprungbefehl für Ingrenz.
A1NACHA2 $C73F - $C74B
Kopiert Akku1 nach Akku2.
INBL $C7A7 - $C7E0
Holt Blocklänge vom Bildschirm nach Akku1.
INID $C7EF - $C814
Holt ID vom Bildschirm nach Akku1.
ORDER $C844 - $C90C
Sortiert aufsteigend den Zeigerstapel, auf den X/Y zeigt, nach den Kriterien, die in A stehen (Bitmaske).
Z1Z3Z2Z4 $C90C - $C91D
Kopiert Zeiger1 nach Zeiger3 und Zeiger2 nach Zeiger4.
CHKSTRDA $CB52 - $CB75
Prüft nach den Kriterien in Bitmaske, ob sich das Element in Akku1 in der Liste befindet und holt gegebenenfalls Adresse nach Zeiger 4.
Gefunden: C=1.
CORPTRSUB $CB9B - $CC5B
Korrigiert alle Zeiger des Zeigerstapels, auf den X/Y zeigt, um den Wert Index, falls der betreffende Zeiger größer als Zeiger3 ist. Falls der betreffende Zeiger = Zeiger3 ist, dann wird dieser aus dem Stapel entfernt.
CORPTRDE $CC5B - $CC64
Decrementiert Zeiger für »Corptrsub«.
CORPTRIM $CC64 - $CC70
Incrementiert Zeiger für »Corptrsub«. Falls Ende erreicht, dann C=0.
CORPTRUE $CC67 - $CC70
Vermindert den Wert für den zu korrigierenden Zeigerstapel um eine Page.
CORPTR $CCFB - $CD10
Korrigiert alle Zeigerstapel. »Corptrsub«.
PARA $CD10 - $CD6D
Holt gesamtes Element vom Bildschirm und setzt entsprechend der verwendeten Joker Bitmaske.

Variable

Zeiger 1 = $61
Zeiger 2 = $63
Zeiger 3 = $65
Zeiger 4 = $FC

Bitmaske $C34C - Bedeutung siehe »Compare«
Namlong1 $C34D - siehe Getlong1
Namlong2 $C34E - siehe Getlong2
Kursor $C34F - siehe Tab.

Akku2 = $CEB0 - $CEC4 -Speicherplatz für Untergrenzwert beim Listen
Akku1 = $CEC4 - $CED8 -Hauptarbeitsspeicher für Fileelemente
Akku3 = $CED8 - $CEEC -Speicher für Obergrenzwert beim Listen

Stackpnt $CE8F -Zwischenspeicher für Stackpointer Speicher $CE90, Speicher2 $CE91, Speicherplatz, um Zwischenwerte abzulegen, die nicht problemlos auf dem Stack zwischengespeichert werden können.

Hzeiger $CE9 2 Hilfszeiger für Zwischenwerte

Index $CE94 wird meist als Zähler verwendet.

Stackbot $CE96 dient zur Übergabe der Anfangsadressen der Zeigerstapel an Unterroutinen.

Ugrenze $CFC7 enthält Adresse auf Anfang Speicherbereich für Filenamen.

Counter $CFD1 enthält Anzahl Namen in Liste
Bpointer $CFD3 zeigt auf Ende Zeigerstapel für Blook
Ipointer $CFD5 zeigt auf Ende Zeigerstapel für Ilook
Apointer $CFD7 zeigt auf Ende Zeigerstapel für Alook
Spointer $CFD9 zeigt auf Ende benutzter Bereich für

Konstante

Ogrenz = $C000 Obergrenze für Speicherplatz Filenamen.
Nammax = 2053 maximale Anzahl Namen in Liste

Datentabellen

ZweiPot $CE98 - $CEA0 enthält 255-Zweierpotenzen
Kurtable $CEA0 - $CEA8 Tabulatorstops für Tab
Comtablo $CEA8 - $CEACLowbytes der Adressen der Vergleichsroutinen
Comtabhi $CEAC - $CEB0 Highbytes -
Newtab $CEEC - $CF99 Schlüsselworttabelle für Keystr.
Newtable $CF47 - $CF99 Schlüsselworttabelle für neue Token
Adrtable $CF99 - $CFBB Adressen-1 der Befehle
Filetyp $CFBB - $CFBF Anfangsbuchstaben der Filetypen
Altwerte $CFBF - $CFC9 Anfangswerte für Counter, Bpointer, Ipointer, Apointer und Spointer
Vecaltta $EBB4 Normale Adressen für Klartext in Token und umgekehrt, Statement ausführen.
Vecneuta $CFC9 - $CFD1 Neue Werte für Klartext in Token....

Anfangsadressen der Befehle

@	$C66A	Quit	$C3AE	Kill	$C3BB
Omerge	$C5E1	Alook	$C74B	Blook	$C7E0
Ilook	$C814	Sort	$C823	Put	$C9AE
Merge	$CA47	Slook	$CB75	Membot	$CC70
Write	$CD6D	Scratch	$CD95	Merge	$CE4E
Help	$CE6A	Mem	$CAF6	Disk	$CAED

```
program.n : file-manager    0801 18fa
--------------------------------------

0801 : 24 08 0a 00 8f 20 2a 20   8f
0809 : 4d 41 44 45 20 49 4e 20   76
0811 : 54 48 45 20 59 45 41 52   48
0819 : 20 31 39 38 35 20 42 59   37
0821 : 20 2a 00 47 08 14 00 8f   80
0829 : 20 2a 20 20 20 20 20 20   2e
0831 : 20 20 20 20 20 20 20 20   31
0839 : 20 20 20 20 20 20 20 20   39
0841 : 20 20 20 20 2a 00 6a 08   da
0849 : 1e 00 8f 20 2a 20 20 20   b4
0851 : 20 20 20 53 20 54 20 4f   b8
0859 : 20 46 20 46 20 45 20 4c   b3
0861 : 20 20 20 20 20 20 20 2a   75
0869 : 00 8d 08 28 00 8f 20 2a   88
0871 : 20 20 20 20 20 20 20 20   71
0879 : 20 20 20 20 20 20 20 20   79
0881 : 20 20 20 20 20 20 20 20   81
0889 : 20 20 2a 00 b0 08 32 00   58
0891 : 8f 20 2a 2a 2a 2a 2a 2a   f1
0899 : 2a 2a 2a 2a 2a 2a 2a 2a   99
08a1 : 2a 2a 2a 2a 2a 2a 2a 2a   a1
08a9 : 2a 2a 2a 2a 2a 2a 00 de   6a
08b1 : 08 3c 00 9e c2 28 34 33   50
08b9 : 29 aa 32 35 36 ac c2 28   8f
08c1 : 34 34 29 aa 32 32 33 3a   a5
08c9 : d2 3a cf c2 28 31 38 36   5e
08d1 : 29 3a d3 34 30 39 36 30   99
08d9 : 3a d4 3a d1 00 00 00 a9   99
08e1 : 00 85 63 a9 cf 85 64 a5   b8
08e9 : 2d 38 e9 ef 85 61 a5 2e   01
08f1 : e9 00 85 62 a2 10 a0 ef   95
08f9 : 88 b1 61 91 63 98 d0 f8   15
0901 : c6 64 c6 62 ca d0 f1 4c   8b
0909 : 50 c3 78 08 48 a9 fc 25   6a
0911 : 01 85 01 68 28 60 08 48   58
0919 : a9 07 05 01 85 01 68 28   fa
0921 : 58 60 ad aa ce 85 63 ad   6c
0929 : ab ce 85 64 06 63 26 64   06
0931 : ad ac ce 18 65 63 85 63   39
0939 : ad ad ce 65 64 85 64 a0   62
0941 : 00 b1 63 85 61 c8 b1 63   8d
0949 : 85 62 60 ad da ce 29 0f   b4
0951 : 8d 4f c3 60 ad c6 ce 29   21
0959 : 0f 8d a2 ce 60 a0 00 b1   20
0961 : 61 20 44 c0 b1 65 20 4d   5d
0969 : c0 a0 04 b1 61 c9 2a f0   a0
0971 : 37 c9 3f f0 10 aa b1 65   63
0979 : c9 2a f0 2c c9 3f f0 05   7d
0981 : 8a d1 65 d0 23 c8 98 38   b3
0989 : e9 05 cd 4f c3 b0 0d cd   e4
0991 : a2 ce 90 d7 b1 61 c9 2a   5b
0999 : f0 0e d0 06 b1 65 c9 2a   47
09a1 : f0 06 ad 4f c3 cd a2 ce   bc
09a9 : 60 a0 00 b1 65 29 30 8d   0b
09b1 : a8 ce b1 61 29 30 cd a8   f5
09b9 : ce d0 05 c8 b1 61 d1 65   82
09c1 : 60 a0 02 b1 61 d1 65 d0   04
09c9 : 05 c8 b1 61 d1 65 60 a0   d6
09d1 : 00 b1 65 29 c0 8d a8 ce   e1
09d9 : b1 61 29 c0 cd a8 ce 60   bb
09e1 : ad 4e c3 8d a6 ce a2 08   d3
09e9 : 0a b0 05 ca d0 fa 38 60   6c
09f1 : e0 05 90 08 ca ca ca ca   3d
09f9 : 29 ef f0 17 8e a7 ce bd   16
0a01 : bd ce 8d 02 c1 bd c1 ce   78
0a09 : 8d 03 c1 20 66 fe d0 df   ee
0a11 : ae a7 ce ad a6 ce 3d ad   2d
0a19 : ce 3d b1 ce 4c d9 c0 a6   b0
0a21 : 7a a0 04 84 0f bd 00 02   60
0a29 : 10 07 c9 ff f0 3e e8 d0   75
0a31 : f4 c9 20 f0 37 85 08 c9   83
0a39 : 22 f0 55 24 0f 70 2d c9   6a
0a41 : 3f d0 04 a9 99 d0 25 c9   67
0a49 : 30 90 04 c9 3c 90 1d 84   c1
0a51 : 71 a0 00 84 0b 88 86 7a   a7
0a59 : ca c8 e8 bd 00 02 38 f9   5e
0a61 : 9e a0 f0 f5 c9 80 d0 2f   8c
0a69 : 05 0b a4 71 e8 c8 99 fb   7e
0a71 : 01 c9 00 f0 38 38 e9 3a   d6
0a79 : f0 06 c9 49 d0 02 85 0f   59
0a81 : 38 e9 55 d0 a0 85 08 bd   ef
0a89 : 00 02 f0 e0 c5 08 f0 dc   fd
0a91 : c8 99 fb 01 e8 d0 f0 a6   6b
0a99 : 7a e6 0b c8 b9 9d a0 10   8d
0aa1 : fa b9 9e a0 d0 b5 f0 0f   d0
0aa9 : bd 00 02 10 bd 99 fd 01   8c
0ab1 : c6 7b a9 ff 85 7a 60 a0   8e
0ab9 : 00 b9 5d cf d0 02 c8 e8   f9
0ac1 : bd 00 02 38 f9 5d cf f0   b2
0ac9 : f5 c9 80 d0 04 05 0b d0   13
0ad1 : 99 a6 7a e6 0b c8 b9 5c   cf
0ad9 : cf 10 fa b9 5d cf d0 e0   ff
0ae1 : f0 c6 a0 ff c8 b9 02 cf   5e
0ae9 : 10 fa ca d0 f7 c8 b9 02   f4
0af1 : cf 30 06 20 dd c2 4c e3   4b
0af9 : c1 29 7f 4c dd c2 c9 cc   6d
0b01 : b0 04 aa 4c 1a a7 c9 dd   a9
0b09 : b0 f8 24 0f 30 f4 38 e9   80
0b11 : c0 aa 84 49 20 d8 c1 a4   8a
0b19 : 49 4c 00 a7 a8 a5 65 48   5b
0b21 : a5 66 48 a5 61 48 a5 62   74
0b29 : 48 98 20 cd bd 98 18 6d   5b
0b31 : a3 ce 8d a3 ce 8c a6 ce   9d
0b39 : 20 d8 c2 ac a6 ce c8 c0   91
0b41 : 06 d0 f2 68 85 62 68 85   91
0b49 : 61 68 85 66 68 85 65 60   15
0b51 : ad da ce 29 30 4a 4a 4a   57
0b59 : 4a ae db ce 20 12 c2 20   a9
0b61 : db c2 20 41 c0 a8 c8 8c   5b
0b69 : a6 ce a0 00 b9 de ce 20   ac
0b71 : dd c2 c8 cc a6 ce d0 f4   89
0b79 : 20 db c2 20 d8 c2 c8 c0   84
0b81 : 11 d0 f8 ad da ce 29 c0   38
0b89 : 18 2a 2a 2a aa e8 20 d8   aa
0b91 : c1 20 a9 c2 20 db c2 ad   6c
0b99 : dc ce 20 dd c2 ad dd ce   4f
0ba1 : 20 dd c2 20 db c2 20 c9   4c
0ba9 : c2 20 e4 ff f0 05 20 e4   36
0bb1 : ff f0 fb a6 9a ad a3 ce   3f
0bb9 : e0 03 f0 03 c9 47 2c c9   d2
0bc1 : 1f b0 1c a2 08 dd b5 ce   78
0bc9 : f0 14 ca d0 f8 20 d8 c2   09
0bd1 : 4c a9 c2 20 e1 ff d0 06   14
0bd9 : 20 42 f6 4c 86 e3 60 a9   be
0be1 : 0d 2c a9 20 2c a9 22 48   9c
0be9 : c9 0d f0 05 ee a3 ce 10   7d
0bf1 : 05 a9 00 8d a3 ce 68 4c   67
0bf9 : d2 ff 8e a6 ce 8c a7 ce   d1
0c01 : a5 fc 38 e5 63 8d a8 ce   d2
0c09 : a5 fd e5 64 8d a9 ce 98   45
0c11 : 30 5e a5 fc 38 ed a8 ce   ac
0c19 : 8d 36 c3 a5 fd e9 00 8d   b1
0c21 : 37 c3 ad 36 c3 18 6d a6   6c
0c29 : ce 8d 39 c3 ad 37 c3 6d   03
0c31 : a7 ce 8d 3a c3 [illegible] [illegible] ce   e0
0c39 : e8 ac a8 ce f0 0a 88 b9   70
0c41 : 00 10 99 00 10 98 d0 f6   a7
0c49 : ce 37 c3 ce 3a c3 ca d0   0c
0c51 : ed 60 20 0c c0 4c 66 fe   fe
0c59 : 08 00 a2 06 bd de cf 9d   18
0c61 : 03 03 ca d0 f7 a9 48 8d   bb
0c69 : fa ff a9 c3 8d fb ff 60   bf
0c71 : a5 63 18 6d a8 ce 8d 9c   ec
0c79 : c3 a5 64 69 ff 8d 9d c3   bf
0c81 : ad 9c c3 18 6d a6 ce 8d   d3
0c89 : 9f c3 ad 9d c3 6d a7 ce   0d
0c91 : 8d a0 c3 a9 00 38 ed a8   5f
0c99 : ce 8d a8 ce ae a9 ce e8   77
0ca1 : ac a8 ce f0 09 b9 00 10   f1
0ca9 : 99 00 10 c8 d0 f7 ee 9d   23
0cb1 : c3 ee a0 c3 ca d0 ee 60   3b
0cb9 : a2 06 bd 4a e4 9d 03 03   64
0cc1 : ca d0 f7 f0 63 a2 0a bd   fe
0cc9 : d4 cf 9d e4 cf ca d0 f7   0f
0cd1 : f0 56 a2 a2 a0 c6 a9 01   d2
0cd9 : 20 bd ff a0 60 a9 00 85   4a
0ce1 : 90 ae a4 ce 20 ba ff 20   e3
0ce9 : e5 c3 a2 05 6c 00 03 20   0c
0cf1 : d5 f3 68 68 a5 ba 20 b4   01
0cf9 : ff a5 b9 4c 96 ff a9 00   d3
0d01 : a0 6f ae a4 ce 20 ba ff   72
0d09 : 85 90 8a 20 b1 ff a5 b9   a2
0d11 : 20 93 ff a5 90 30 d3 60   4a
0d19 : a4 90 d0 03 4c a5 ff 20   cc
0d21 : ab ff 20 42 f6 20 25 c4   ab
0d29 : ae a5 ce 9a 4c ae a7 20   ca
0d31 : f4 c3 20 ab ff 20 ea c3   b8
0d39 : 20 a5 ff 20 dd c2 c9 0d   65
0d41 : d0 f6 20 ab ff 4c 42 f6   63
0d49 : a6 22 a4 23 ad a6 ce 20   19
0d51 : ce c3 a0 05 8c a6 ce 20   43
0d59 : 0e c4 ce a6 ce d0 f8 8d   c4
0d61 : a7 ce 20 0e c4 ae a7 ce   37
0d69 : 20 cd bd a9 20 20 dd c2   14
0d71 : 20 0e c4 aa d0 f7 20 d5   18
0d79 : c2 20 c9 c2 a0 03 d0 d4   25
0d81 : ad ed cf 38 ed dd cf 8d   48
0d89 : a8 ce ad ee cf ed de cf   69
0d91 : 8d a9 ce 60 a9 00 38 ed   0a
0d99 : ed cf 85 63 a9 c0 ed ee   72
0da1 : cf 85 64 60 a9 00 38 ed   af
0da9 : e5 cf 85 fc a9 08 ed e6   d7
0db1 : cf 85 fd 60 a2 00 86 62   d7
0db9 : 86 66 e8 86 65 20 41 c0   5b
0dc1 : 18 69 05 85 61 20 8a c4   4b
0dc9 : a5 64 c5 62 b0 06 20 42   9e
0dd1 : f6 4c 35 a4 d0 06 a5 63   6a
0dd9 : c5 61 90 f2 20 9a c4 a5   06
0de1 : fd c5 66 90 e9 d0 06 a5   f5
0de9 : fc c5 65 90 e1 60 20 aa   2a
0df1 : c4 20 00 c0 ad eb cf 85   62
0df9 : 61 ad ec cf 85 62 ad e7   58
0e01 : cf 85 63 ad e8 cf 85 64   0d
0e09 : ad e9 cf 85 65 ad ea cf   5e
0e11 : 85 66 20 41 c0 18 69 05   76
0e19 : 8d a2 ce a0 00 ad ed cf   84
0e21 : 85 fc 91 61 91 63 91 65   fa
0e29 : 18 6d a2 ce 8d ed cf c8   93
0e31 : ad ee cf 85 fd 91 61 91   0f
0e39 : 63 91 65 90 03 ee ee cf   d3
0e41 : 88 b9 da ce 91 fc c8 cc   f4
0e49 : a2 ce d0 f5 ee e5 cf d0   44
0e51 : 03 ee e6 cf ad eb cf 18   28
0e59 : 69 02 8d eb cf 90 03 ee   0f
0e61 : ec cf ad e7 cf 18 69 02   04
0e69 : 8d e7 cf 90 03 ee e8 cf   db
0e71 : ad e9 cf 18 69 02 8d e9   ba
0e79 : cf 90 03 ee ea cf 4c 0c   a5
0e81 : c0 20 c8 c3 a0 1b 8c a6   5e
0e89 : ce 20 0e c4 ce a6 ce d0   82
0e91 : f8 8d dc ce 20 0e c4 8d   01
0e99 : dd ce 20 0e c4 aa d0 fa   82
0ea1 : 20 0e c4 20 0e c4 20 0e   a1
0ea9 : c4 8d db ce 20 0e c4 29   dc
0eb1 : 03 0a 0a 0a 0a 8d da ce   93
0eb9 : 20 0e c4 c9 22 d0 f9 a2   20
0ec1 : 00 8e a6 ce 20 0e c4 ae   6f
0ec9 : a6 ce c9 22 f0 06 9d de   00
0ed1 : ce e8 d0 ed ca 8a 0d da   f0
0ed9 : ce 8d da ce 8e a6 ce 20   98
0ee1 : 0e c4 ae a6 ce e8 e0 10   aa
0ee9 : d0 f2 c9 2a f0 ac 20 0e   fb
0ef1 : c4 a2 04 dd d0 cf f0 05   1c
0ef9 : ca d0 f8 f0 9d ca 8a 18   12
0f01 : 6a 6a 6a 0d da ce 8d da   ed
0f09 : ce 60 20 77 c5 20 e4 c4   79
0f11 : 20 90 c5 4c 03 c6 a2 14   8d
0f19 : a9 00 9d c5 ce ca d0 fa   5f
0f21 : a2 14 a9 ff 9d ed ce ca   52
0f29 : d0 f8 60 a9 ff 8d aa ce   77
0f31 : 8d ab ce 60 ee aa ce d0   75
0f39 : 03 ee ab ce ad aa ce cd   7f
0f41 : e5 cf f0 01 60 ad ab ce   2a
0f49 : cd e6 cf 60 a0 00 b1 61   1d
0f51 : 20 44 c0 18 69 05 8d a2   01
0f59 : ce b1 61 99 da ce c8 cc   6c
0f61 : a2 ce d0 f5 60 20 73 00   32
0f69 : c9 40 f0 27 c9 cc b0 06   45
0f71 : 20 79 00 4c e7 a7 c9 dd   76
0f79 : b0 f6 ba 8e a5 ce 38 e9   aa
0f81 : cc 0a aa a9 00 8d a3 ce   cb
0f89 : bd b0 cf 48 bd af cf 48   c4
0f91 : 4c 73 00 20 73 00 d0 03   1b
0f99 : 4c 1b c4 20 9e ad 20 a3   c7
0fa1 : b6 8d a6 ce aa f0 0e a0   4d
0fa9 : 00 b1 22 c9 24 d0 03 4c   b1
0fb1 : 3e c4 20 ae c6 4c 1e c4   00
0fb9 : 20 f4 c3 a0 00 8c a7 ce   f9
0fc1 : b1 22 20 a8 ff ac a7 ce   42
0fc9 : c8 cc a6 ce d0 ef 4c ae   96
0fd1 : ff a5 7a d0 02 c6 7b c6   2d
0fd9 : 7a 60 20 79 00 c9 ac d0   5d
0fe1 : 05 20 73 00 18 60 20 c7   68
0fe9 : c6 20 86 ae 20 a3 b6 8d   4c
0ff1 : a6 ce aa d0 05 a2 08 4c   e1
0ff9 : 8b e3 a0 00 b1 22 99 de   ee
1001 : ce c8 c0 10 f0 05 cc a6   1d
1009 : ce d0 f1 88 8c a6 ce ad   61
1011 : da ce 29 f0 0d a6 ce 8d   17
1019 : da ce 38 60 20 0c c6 20   32
1021 : 79 00 f0 14 c9 ab f0 18   47
1029 : 20 4c c7 90 0b 20 4f c7   f2
1031 : 20 79 00 d0 04 20 40 c7   fa
1039 : 60 c9 ab f0 03 4c 08 af   99
1041 : 20 73 00 f0 f3 20 4c c7   3a
1049 : 90 ee a2 14 bd d9 ce 9d   9d
1051 : ed ce ca d0 f7 60 4c 66   f2
1059 : fe a2 14 bd d9 ce 9d c5   7b
1061 : ce ca d0 f7 60 a9 d0 48   ee
1069 : a9 c6 48 a2 f8 a0 ef a9   83
1071 : 08 8d 4e c3 8e ac ce 8c   ee
```

Listing. »Filemanager« bringt Ordnung in Ihre Diskettensammlung

```
1079 : ad ce 68 8d 4e c7 68 8d  39
1081 : 4d c7 20 12 c7 20 21 c6  8c
1089 : 20 00 c0 20 2a c6 d0 06  06
1091 : 20 0c c0 4c 1e c4 20 18  2a
1099 : c0 a9 c6 85 65 a9 ce 85  7a
10a1 : 66 20 d6 c0 90 e2 a9 ee  8a
10a9 : 85 65 a9 ce 85 66 20 d6  df
10b1 : c0 f0 02 b0 d3 20 42 c6  55
10b9 : 20 0c c0 20 46 c2 4c 7e  bc
10c1 : c7 20 79 00 c9 ac d0 05  46
10c9 : 20 73 00 18 60 20 c7 c6  5a
10d1 : 20 86 ae 20 f7 b7 a5 15  e2
10d9 : c9 04 90 03 4c 48 b2 0a  0f
10e1 : 0a 0a 0a 48 a9 cf 2d da  ff
10e9 : ce 8d da ce 68 0d da ce  06
10f1 : 8d da ce a5 14 8d db ce  0e
10f9 : 38 60 a9 b7 48 a9 c7 48  44
1101 : a2 f8 a0 cf a9 04 4c 67  fc
1109 : c7 20 79 00 c9 ac d0 05  8e
1111 : 20 73 00 18 60 20 c7 c6  a2
1119 : 20 86 ae 20 a3 b6 aa f0  a8
1121 : f2 a0 00 b1 22 99 dc ce  9a
1129 : c8 c0 02 d0 f6 60 a9 ff  05
1131 : 48 a9 c7 48 a2 f8 a0 df  7d
1139 : a9 02 4c 67 c7 20 00 c0  e2
1141 : a2 f8 a0 ef a9 8f 20 54  c6
1149 : c8 a2 f8 a0 cf a9 4f 20  7c
1151 : 54 c8 a2 f8 a0 df a9 2f  df
1159 : 20 54 c8 4c 86 c7 8d 4e  d8
1161 : c3 8e ac ce 8c ad ce 20  22
1169 : 21 c6 20 2a c6 d0 01 60  f2
1171 : 20 18 c0 20 1c c9 a5 62  3d
1179 : 20 2a c6 f0 f2 20 18 c0  90
1181 : 20 d6 c0 b0 ee ad aa ce  f7
1189 : 8d da ce 8d de ce ad ab  5b
1191 : ce 8d db ce 8d df ce a9  5d
1199 : 00 8d dc ce 8d dd ce 20  b4
11a1 : 1c c9 ad dc ce 18 6d de  ca
11a9 : ce 8d aa ce ad dd ce 6d  a2
11b1 : df ce 8d ab ce 4e ab ce  7c
11b9 : 6e aa ce 20 18 c0 20 d6  ea
11c1 : c0 f0 33 08 ad dd ce cd  68
11c9 : ab ce d0 08 ad dc ce cd  a9
11d1 : aa ce f0 19 ad aa ce ae  0a
11d9 : ab ce 28 90 08 8d de ce  0d
11e1 : 8e df ce b0 bd 8d dc ce  82
11e9 : 8e dd ce 90 b5 28 b0 06  97
11f1 : 20 2a c6 20 18 c0 a0 00  e6
11f9 : b1 fc 8d e0 ce c8 b1 fc  9c
1201 : 8d e1 ce a0 00 a2 02 20  a4
1209 : f0 c2 a0 00 ad e0 ce 91  c3
1211 : 63 c8 ad e1 ce 91 63 ad  e2
1219 : db ce 8d ab ce ad da ce  97
1221 : 8d aa ce 4c 66 c8 a5 61  46
1229 : 85 65 a5 62 85 66 a5 63  ff
1231 : 85 fc a5 64 85 fd 60 ad  4f
1239 : dd cf 85 fc ad de cf 85  1b
1241 : fd 4c 4b c9 a0 00 b1 fc  3b
1249 : 29 0f 18 69 05 65 fc 85  a8
1251 : fc 90 02 e6 fd a5 fc cd  8f
1259 : ed cf d0 05 a5 fd cd ee  62
1261 : cf 60 20 9e ad 20 a3 b6  14
1269 : 8d a6 ce 20 f4 c3 a9 53  bc
1271 : 20 a8 ff a9 3a 20 a8 ff  62
1279 : 20 b1 c6 20 25 c4 a6 22  7f
1281 : a4 23 ad a6 ce 20 bd ff  dc
1289 : a9 00 a0 61 ae a4 ce 20  12
1291 : ba ff 20 d5 f3 90 03 4c  76
1299 : e0 c3 a5 ba 20 b1 ff a5  f6
12a1 : b9 20 93 ff 20 76 c4 ad  73
12a9 : a8 ce 20 a8 ff ad a9 ce  87
12b1 : 20 a8 ff ad e5 cf 20 a8  89
12b9 : ff ad e6 cf 20 a8 ff 20  ca
12c1 : 21 c6 20 00 c0 20 2d c9  a3
12c9 : 20 2a c6 d0 09 20 0c c0  0d
12d1 : 20 ae ff 4c 1b c4 a5 fc  3a
12d9 : 85 61 a5 fd 85 62 20 42  a8
12e1 : c6 a9 f8 8d ac ce a9 ef  33
12e9 : 8d ad ce 20 18 c0 a5 61  e5
12f1 : 38 ed dd cf 8d d4 ce a5  97
12f9 : 62 ed de cf 8d d5 ce a9  19
1301 : f8 8d ac ce a9 cf 8d ad  6f
1309 : ce 20 18 c0 a5 61 38 ed  28
1311 : dd cf 8d d6 ce a5 62 ed  93
1319 : de cf 8d d7 ce a9 f8 8d  76
1321 : ac ce a9 df 8d ad ce 20  5c
1329 : 18 c0 a5 61 38 ed dd cf  41
1331 : 8d d8 ce a5 62 ed de cf  43
1339 : 8d d9 ce 20 41 c0 18 69  b8
1341 : 0b 8d a2 ce 20 0c c0 a0  3c
1349 : 00 8c a6 ce b9 d4 ce 20  d0
1351 : a8 ff ac a6 ce c8 cc a2  a4
1359 : ce d0 ee 20 00 c0 20 3a  4a
1361 : c9 4c be c9 20 9e ad 20  27
1369 : a3 b6 a6 22 a4 23 20 ce  d7
1371 : c3 20 0e c4 85 61 20 0e  60
1379 : c4 85 62 20 0e c4 85 65  84
1381 : 20 0e c4 85 66 20 bb c4  6a
1389 : ad ed cf 8d a8 ce ad ee  68
1391 : cf 8d a9 ce a0 00 8c a6  f4
1399 : ce 20 0e c4 ac a6 ce 99  02
13a1 : d4 ce c8 c0 07 d0 ef 20  1d
13a9 : 41 c0 18 69 05 8d a2 ce  62
13b1 : a0 01 8c a6 ce 20 0e c4  79
13b9 : ac a6 ce 99 da ce c8 cc  80
13c1 : a2 ce d0 ee 20 e4 c4 20  59
13c9 : 00 c0 a0 00 ad d4 ce 18  3e
13d1 : 6d a8 ce 91 61 c8 ad d5  37
13d9 : ce 6d a9 ce 91 61 88 ad  44
13e1 : d6 ce 18 6d a8 ce 91 63  e0
13e9 : c8 ad d7 ce 6d a9 ce 91  da
13f1 : 63 88 ad d8 ce 18 6d a8  d3
13f9 : ce 91 65 c8 ad d9 ce 6d  c2
1401 : a9 ce 91 65 20 0c c0 4c  20
1409 : 8a ca 20 9e b7 8e a4 ce  f4
1411 : 4c 1e c4 20 76 c4 ad e5  b1
1419 : cf ac e6 cf a2 05 20 41  47
1421 : cb 20 8a c4 a5 63 8d a8  34
1429 : ce a5 64 8d a9 ce 20 9a  5b
1431 : c4 a5 fc a4 fd a2 06 20  e9
1439 : 41 cb a2 13 20 d8 c1 ae  98
1441 : dd cf ad de cf 20 12 c2  19
1449 : 4c 1e c4 85 61 84 62 20  8a
1451 : d8 c1 ae a8 ce ad a9 ce  69
1459 : 20 12 c2 a2 07 20 d8 c1  e0
1461 : a6 61 a5 62 20 12 c2 a2  50
1469 : 08 20 d8 c1 4c d5 c2 a9  c1
1471 : da 85 65 a9 ce 85 66 20  8f
1479 : 2d c9 f0 14 a5 fc 85 61  65
1481 : a5 fd 85 62 20 d6 c0 d0  30
1489 : 02 38 60 20 3a c9 d0 ec  d3
1491 : 18 60 a9 0d 8d 4e c3 20  80
1499 : 77 c5 20 00 c0 20 65 cb  35
14a1 : 20 0c c0 90 05 a9 12 20  30
14a9 : d2 ff 20 46 c2 a9 92 20  50
14b1 : d2 ff 20 90 c5 4c 90 cb  36
14b9 : 86 63 84 64 ad e5 cf 8d  03
14c1 : a8 ce ad e6 cf [illegible] a9 ce  c6
14c9 : 0e a8 ce 2e a9 ce a5 63  13
14d1 : 18 6d a8 ce 85 fc a5 64  43
14d9 : 6d a9 ce 85 fd a5 63 18  4a
14e1 : 6d a8 ce 85 63 b0 02 c6  58
14e9 : 64 a9 00 38 ed a8 ce 8d  a3
14f1 : a8 ce ae a9 ce e8 ac a8  19
14f9 : ce d0 05 20 7a cc 90 25  0f
1501 : b1 63 8d a8 ce 20 77 cc  42
1509 : 90 1b b1 63 8d a9 ce c5  ec
1511 : 66 90 0d f0 02 b0 0f ad  5e
1519 : a8 ce c5 65 f0 22 b0 06  35
1521 : 20 77 cc b0 db 60 20 6e  64
1529 : cc ad a8 ce 18 6d aa ce  05
1531 : 91 63 ad a9 ce 6d ab ce  b9
1539 : 20 77 cc 91 63 4c 16 cc  05
1541 : a5 63 48 a5 64 48 8a 48  a2
1549 : 98 48 20 77 cc b0 05 68  34
1551 : 68 68 68 60 98 18 65 63  ba
1559 : 85 63 90 02 e6 64 a0 ff  08
1561 : a2 fe 20 f0 c2 68 a8 68  8b
1569 : aa 68 85 64 68 85 63 20  b6
1571 : 6e cc 20 6e cc 4c 16 cc  3c
1579 : c0 00 d0 03 c6 64 e8 88  12
1581 : 60 c8 d0 07 e6 64 ca d0  b9
1589 : 02 18 60 38 60 20 8a ad  43
1591 : 20 f7 b7 20 76 c4 a5 14  eb
1599 : 18 6d a8 ce 85 61 a5 15  90
15a1 : 6d a9 ce 85 62 b0 06 c9  9e
15a9 : c0 90 0b f0 03 4c 35 a4  43
15b1 : a9 00 c5 61 90 f7 a5 15  81
15b9 : c5 38 90 f1 d0 06 a5 14  f9
15c1 : c5 37 90 e9 a5 14 38 ed  3b
15c9 : dd cf 8d aa ce a5 15 ed  91
15d1 : de cf 8d ab ce a9 00 85  b5
15d9 : 65 85 66 20 00 c0 ad dd  17
15e1 : cf 85 63 ad de cf 85 64  4d
15e9 : ad ed cf 85 fc ad ee cf  ca
15f1 : 85 fd ae aa ce ac ab ce  14
15f9 : 20 f0 c2 20 0e cd 20 0c  2e
1601 : c0 a5 14 8d dd cf a5 15  68
1609 : 8d de cf a5 61 8d ed cf  88
1611 : a5 62 8d ee cf 4c 1e c4  8a
1619 : a2 f8 a0 ef 20 ae cb a2  49
1621 : f8 a0 cf 20 ae cb a2 f8  27
1629 : a0 df 4c ae cb a9 00 8d  c7
1631 : da ce 20 b7 c7 90 03 a9  d2
1639 : 0f 2c a9 0b 48 20 d0 c6  80
1641 : b0 04 68 29 07 48 20 79  59
1649 : 00 a2 04 dd d0 cf f0 14  cf
1651 : ca d0 f8 c9 b7 f0 1c c9  02
1659 : ac f0 03 4c 08 af 68 29  ba
1661 : 0e 48 d0 0f 86 61 a2 02  ab
1669 : 20 93 ce f0 17 ca d0 f8  21
1671 : a6 61 2c a2 03 20 f3 c5  b4
1679 : 20 73 00 f0 07 20 ff c7  72
1681 : 68 b0 04 48 68 29 0d 8d  6b
1689 : 4e c3 60 20 23 cd c9 0f  bb
1691 : d0 19 20 41 c0 a8 c8 b9  06
1699 : dd ce c9 2a f0 0d c9 3f  b2
16a1 : f0 09 88 d0 f2 20 e4 c4  9f
16a9 : 4c 1e c4 a2 09 20 d8 c1  02
16b1 : 4c 86 e3 20 23 cd 20 00  5e
16b9 : c0 a9 00 8d 36 c3 8d 37  26
16c1 : c3 20 65 cb b0 14 ad 37  38
16c9 : c3 ae 36 c3 20 0c c0 20  8f
16d1 : 12 c2 a2 0a 20 d8 c1 4c  97
16d9 : 1e c4 ee 36 c3 d0 03 ee  88
16e1 : 37 c3 a5 fc 85 65 a5 fd  19
16e9 : 85 66 a0 ff 8c ab ce c8  bc
16f1 : b1 fc 20 44 c0 18 69 05  2d
16f9 : 8d a2 ce 49 ff 8d aa ce  69
1701 : ee aa ce 20 0e cd a5 65  ad
1709 : 18 6d a2 ce 85 63 a5 66  31
1711 : 69 00 85 64 ad ed cf 85  fd
1719 : fc 38 ed a2 ce 8d ed cf  b2
1721 : ad ee cf 85 fd e9 00 8d  34
1729 : ee cf ae aa ce ac ab ce  9e
1731 : 20 f0 c2 ad e5 cf d0 03  56
1739 : ce e6 cf ce e5 cf ad eb  b3
1741 : cf 38 e9 02 8d eb cf b0  c0
1749 : 03 ce ec cf ad e7 cf 38  b2
1751 : e9 02 8d e7 cf b0 03 ce  c8
1759 : e8 cf ad e9 cf 38 e9 02  3c
1761 : 8d e9 cf b0 03 ce ea cf  df
1769 : 4c b7 cd a9 0d 8d 4e c3  37
1771 : 20 77 c5 20 00 c0 20 65  14
1779 : cb 20 0c c0 b0 03 20 e4  dd
1781 : c4 20 90 c5 4c 69 ce a2  c3
1789 : 0b 8e a6 ce 20 d8 c1 20  6f
1791 : a9 c2 ae a6 ce e8 e0 1d  0e
1799 : d0 ef 4c 1e c4 e6 7a d0  47
17a1 : 02 e6 7b a0 00 b1 7a f0  63
17a9 : 02 c9 3a 60 01 00 08 fa  51
17b1 : 05 04 00 20 60 05 f8 ef  ae
17b9 : fe fd fb f7 ef df bf 7f  b0
17c1 : 00 0a 14 1e 28 32 3c 46  21
17c9 : c5 b7 9f 53 c0 c0 c0 c0  53
17d1 : 01 ca 33 32 54 2a 52 42  af
17d9 : 4f 20 4c 4f 4f 52 54 4d  a9
17e1 : 41 4e 53 41 06 08 53 41  b7
17e9 : 54 55 52 4b 45 59 20 4f  24
17f1 : 20 20 4e 4c 46 20 4d 41  5c
17f9 : 01 ca 33 32 54 2a 52 42  d7
1801 : 4f 20 4c 4f 4f 52 54 4d  d1
1809 : 41 4e 53 41 ff 50 52 c7  ca
1811 : 53 45 d1 55 53 d2 52 45  c5
1819 : cc 46 55 4c 4c 3a a0 46  8d
1821 : 52 45 45 3a a0 42 59 54  d9
1829 : 45 53 20 3d a0 4e 41 4d  e4
1831 : 45 d3 3f 20 49 4c 4c 45  e6
1839 : 47 41 4c 20 4e 41 4d 45  e7
1841 : 20 45 52 52 4f d2 4e 41  2a
1849 : 4d 45 53 20 53 43 52 41  2d
1851 : 54 43 48 45 c4 54 48 45  9c
1859 : 20 4b 45 59 57 4f 52 44  5d
1861 : 53 20 41 52 45 3a 8d 41  3e
1869 : 4c 4f 4f cb 42 4c 4f 4f  0c
1871 : cb 49 4c 4f 4f cb 44 49  d5
1879 : 53 cb 44 4d 45 52 47 c5  fc
1881 : 48 45 4c d0 4b 49 4c cc  63
1889 : 4d 45 4d 42 4f d4 4d 45  70
1891 : cd 4d 45 52 47 c5 50 55  2f
1899 : d4 51 55 49 d4 53 43 52  2e
18a1 : 41 54 43 c8 53 4c 4f 4f  6a
18a9 : cb 53 4d 45 52 47 c5 53  37
18b1 : 4f 52 d4 57 52 49 54 c5  96
18b9 : 00 5a c7 ef c7 23 c8 ff  8f
18c1 : ca ff c5 7c ce ba c3 82  63
18c9 : cc 08 cb 59 ca 57 c9 ad  a1
18d1 : c3 a8 cd 87 cb 60 ce 32  ac
18d9 : c8 80 cd 50 53 55 52 00  88
18e1 : 00 f8 cf f8 df f8 ef 00  f6
18e9 : a0 15 c1 f4 c1 5b c6 00  35
18f1 : 00 f8 cf f8 df f8 ef 00  06
18f9 : a0 18 18 18 18 18 18 18  81
```

Listing. »Filemanager« (Schluß)

Vier Bildschirme im Speicher

100 Zeilen Listing auf Tastendruck abrufbereit, vier Bildschirmseiten auf einmal, freie Verwendung in eigenen Programmen. Dies sind die unverkennbaren Vorzüge von »SCREENCOPY«, das sich gleichzeitig als Editierhilfe und Maskenutility präsentiert.

Das Programm »SCREENCOPY« gibt dem Anwender die Möglichkeit, bis zu vier Bildschirmseiten auf Tastendruck in das RAM unter dem Interpreter-ROM ($A000-$BFFF) zu verschieben, wieder in den aktuellen Bildschirm zu holen, oder auf Diskette zu speichern. Hierbei werden alle Farbinformationen (sowie Reverse-Darstellungen etc.) mitgespeichert. Ein Vorteil dieser Speicherbelegung ist, daß die vier Bildschirmseiten keinen Basic-Speicher belegen. »SCREENCOPY« bietet mehrere Anwendungsmöglichkeiten.

1. Beim Editieren von Programmen kann es sehr nützlich sein, wenn man wichtige Notizen schnell auf eine Bildschirmseite schreibt und diese zwischenspeichert. Dies ist erheblich übersichtlicher als herumliegende Notizzettel, da die Informationen auf Tastendruck wieder zur Verfügung stehen. Oder kennen Sie auch folgendes Problem: Man listet das zu bearbeitende Programm und möchte dann einen Programmabschnitt bearbeiten, der sich über mehrere Bildschirmseiten erstreckt. Und dann wird gelistet, verbessert, gelistet.....gelistet. Hier ist die Lösung: Listen Sie die ersten 25 Zeilen und drücken danach auf die F1-Taste. Schon ist der erste Bildschirm gespeichert. Dasselbe passiert mit den jeweils folgenden 25 Zeilen, bis alle vier Funktionstasten mit je einem Bild belegt sind. Somit haben Sie einen Bildschirm mit 100 (hundert!) Zeilen zur Verfügung.

2. Man benützt das Programm zur Erstellung und Planung von Bildschirmmasken (Menüs, Untermenüs etc.). Ist die Maske (oder Masken) fertiggestellt, kann sie mittels CTRL- und entsprechender Funktionstaste auf Diskette gespeichert werden. Das besondere hierbei ist, daß nicht nur ein Bildschirm, sondern zwei, drei oder vier Bildschirme zusammen (unter einem Namen) gespeichert werden können. Um eine hohe Anwenderfreundlichkeit zu realisieren, wurden alle Funktionen auf die Funktionstasten (F1-F8) gelegt. Um die Bildschirmseiten auch innerhalb von Programmen verwenden zu können, steht Ihnen das Programm »PAGE4« zur Verfügung. Der Aufruf der Bildschirme erfolgt über den SYS-Befehl. Das Programm »SCR.RENAME« eignet sich zur Änderung (vor dem Speichern!) der von »SCREENCOPY« festgelegten Bildschirmnamen.

Bedienungsanleitung

SCREENCOPY

Tippen Sie das Programm aus Listing 1 mit dem MSE ab und speichern Sie es auf Diskette. Nach dem absoluten Laden (load "SCREENCOPY",8,1) geben Sie den Befehl »NEW« ein, um ein »OUT OF MEMORY« zu vermeiden. Gestartet wird es mit »SYS 49152«. Jetzt wird bei jedem Interrupt die Abfrageroutine der Funktionstasten durchlaufen. Die Funktionstasten sind wie folgt belegt:

F1 – der aktuelle Bildschirm wird in den Bereich ab $A000 (Seite 1) kopiert.
F3 – der aktuelle Bildschirm wird in den Bereich ab $A800 (Seite 2) kopiert.
F5 – der aktuelle Bildschirm wird in den Bereich ab $B000 (Seite 3) kopiert.
F7 – der aktuelle Bildschirm wird in den Bereich ab $B800 (Seite 4) kopiert.

Bevor Sie den aktuellen Bildschirm mittels F-Tasten unter das ROM kopieren, sollte die Cursorfarbe der Hintergrundfarbe angeglichen werden, da es sonst passieren kann, daß der Cursor mitkopiert wird.

Möchten Sie den Bildschirm (unter dem ROM) wieder in den aktuellen Bildschirm verschieben, dann drücken Sie einfach die Taste »SHIFT« in Kombination mit der gewählten Funktionstaste (das entspricht den Tasten F2/F4/F6/F8).

Nun zum Speichern:

CTRL/F1 – Wenn Sie diese Kombination drücken, wird die Seite 1 auf Diskette gespeichert.
CTRL/F3 – Hier werden Seite 1 und 2 gespeichert.
CTRL/F5 – Bildschirmseiten 1 bis 3 werden gespeichert.
CTRL/F7 – Bildschirmseiten 1 bis 4 werden gespeichert.

Wichtiger Hinweis: Wollen Sie jeden Bildschirm einzeln speichern, so ist das nur über die Kombination CTRL/F1 möglich. Dabei ist zu beachten (wenn schon ein File »SCREEN 1« existiert), daß vor dem Speichern entweder mit dem beiliegenden Programm »SCR.RENAME« der Bildschirmname geändert wird, oder aber der vorhandene Bildschirm auf Diskette mit »OPEN1,8,15,"R:NEUER NAME=ALTER NAME:CLOSE 1« geändert wird, da sonst die Fehlermeldung »FILE EXISTS« auftritt.

PAGE4

Tippen Sie »PAGE4« (Listing 2) mit dem MSE ab. Um die Bildschirmseiten in Programmen zu verwenden, laden Sie das Programm »PAGE4« absolut (,8,1) und geben den Befehl »NEW« ein. Eine Initialisierung ist nicht erforderlich. Wenn PAGE4 von einem Basic-Programm nachgeladen werden soll, gehen Sie wie folgt vor:

1) Laden Sie das Programm, welches »PAGE4« nachladen soll.
2) Geben Sie »PRINT PEEK (45)« und »PRINT PEEK (46)« ein und notieren die angezeigten Werte.
3) Am Anfang des Basic-Programms soll dann folgendes stehen:

```
10 IF A = 0 THEN A = 1:LOAD "PAGE4",8,1
15 IF A = 1 THEN A = 2:LOAD "...",8,1
```

Nachladen der gespeicherten Bildschirme

```
20 POKE 45,X: POKE 46,Y : CLR
```

Die Werte X/Y sind die Zahlen, die in den Zellen (45) und (46) standen.

Der Aufruf der gewünschten Seite erfolgt durch: SYS 828,n (n = erste bis vierte Bildschirmseite). Das Programm »SCREENCOPY« wird natürlich nicht mehr benötigt.

SCR.RENAME

Mit diesem Programm (Listing 3) kann man die Bildschirmnamen *vor* der Speicherung ändern. Zu beachten ist hier, daß der Name max. 8 Zeichen lang sein darf, da die SAVE-Routine mit diesem Wert arbeitet. Der Bildschirmname 1 kann mit diesem Programm direkt geändert werden. Bei Änderung von Namen 2 muß die Variable in der Zeile 65 in

»N2« umgeändert werden. Entsprechendes gilt für Name3 – »N3«, Name4 – »N4«. Es gibt noch 2 Alternativen:
1) Ändern der Namen auf Diskette
2) Die Namen mit einem Maschinensprachenmonitor überschreiben. Sie stehen ab der Adresse $C217 (49687) im Arbeits-Speicher.

Programmbeschreibung:

a) SCREENCOPY: Das Programm »SCREENCOPY« (Listing 4) belegt den Speicherbereich $C000(49152)-$C237(49719). Der Interrupt-Vektor $314/15 zeigt auf die Tastaturabfrage der Routine. Die Konstante $A000 steht in der Zeropage-Adresse $B3/4. Sie dient als Zeiger der Startadresse des zu speichernden Bereiches. Die Zelle $02 wird als Zähler benutzt, damit bei gedrückter Kombination CTRL/F-Taste nicht bei jedem Interrupt die SAVE-Routine angesprungen wird. In den Zellen $FB-$FE werden jeweils die Start- und Endadressen für die Kopier-Routine übergeben.

b) PAGE4: Das Programm »PAGE4« (Listing 5) liegt im Bereich $033C(828) bis $03EE(1066). Hier wird mit voller Absicht mit dem SYS-Befehl gearbeitet, damit keine Vektoren geändert werden. Es verträgt sich mit allen Programmen (Programmerweiterungen), die den Kassettenpuffer nicht für sich beanspruchen. Beim SYS-Aufruf wird zuerst auf nachfolgendes Komma geprüft und der Wert geholt. Ist dieser größer als 4, wird die Fehlermedlung »ILLEGAL QUANTITY« ausgegeben.

c) SCR.RENAME: Die vordefinierten Bildschirmnamen sind:
Seite 1: »SCREEN-1«, Seite 2: »SCREEN-2«
Seite 3: »SCREEN-3«,Seite 4: »SCREEN-4«
N1: Adresse von Name 1($C217-49687)
N2: Adresse von Name 2
N3: Adresse von Name 3
N4: Adresse von Name 4
N$: Bildschirmname
N: Länge des Bildschirmnamens
I: Laufvariable bei der For-Next-Schleife

(W.Schneider/og)

```
programm : screencopy.obj c000 c237
-----------------------------------

c000 : 78 a9 19 a0 c0 8d 14 03   76
c008 : 8c 15 03 58 a9 00 a0 a0   49
c010 : 85 b3 84 b4 a9 05 85 02   03
c018 : 60 ad 8d 02 4a b0 19 4a   16
c020 : 4a b0 18 a5 cb c9 04 f0   7a
c028 : 1e c9 05 f0 2e c9 06 f0   b5
c030 : 3e c9 03 f0 4e 4c 31 ea   14
c038 : 4c b3 c0 c6 02 a5 02 f0   9e
c040 : 03 4c 31 ea 4c 53 c1 20   ba
c048 : 97 c0 85 fd a0 a0 84 fe   7f
c050 : 20 04 c2 20 a0 c0 a0 a4   03
c058 : 4c a9 c0 20 97 c0 85 fd   3e
c060 : a0 a8 84 fe 20 04 c2 20   c3
c068 : a0 c0 a0 ac 4c a9 c0 20   7b
c070 : 97 c0 85 fd a0 b0 84 fe   28
c078 : 20 04 c2 20 a0 c0 a0 b4   4b
c080 : 4c a9 c0 20 97 c0 85 fd   66
c088 : a0 b8 84 fe 20 04 c2 20   f3
c090 : a0 c0 a0 bc 4c a9 c0 a9   b8
c098 : 00 a0 04 85 fb 84 fc 60   33
c0a0 : a9 00 a0 d8 85 fb 84 fc   d1
c0a8 : 60 85 fd 84 fe 20 04 c2   61
c0b0 : 4c 31 ea a5 cb c9 04 f0   01
c0b8 : 0f c9 05 f0 26 c9 06 f0   b6
c0c0 : 3d c9 03 f0 54 4c 31 ea   03
c0c8 : a5 01 48 a9 36 85 01 20   09
c0d0 : 34 c1 85 fb a0 a0 84 fc   e1
c0d8 : 20 04 c2 20 3d c1 a0 a4   5d
c0e0 : 4c 46 c1 a5 01 48 a9 36   da
c0e8 : 85 01 20 34 c1 85 fb a0   f6
c0f0 : a8 84 fc 20 04 c2 20 3d   6f
c0f8 : c1 a0 ac 4c 46 c1 a5 01   c9
c100 : 48 a9 36 85 01 20 34 c1   c1
c108 : 85 fb a0 b0 84 fc 20 04   82
c110 : c2 20 3d c1 a0 b4 4c 46   d7
c118 : c1 a5 01 48 a9 36 85 01   5a
c120 : 20 34 c1 85 fb [illegible] b8 84   2c
c128 : fc 20 04 c2 20 3d c1 a0   c2
c130 : bc 4c 46 c1 a9 00 a0 04   01
c138 : 85 fd 84 fe 60 a9 00 a0   51
c140 : d8 85 fd 84 fe 60 85 fb   ec
c148 : 84 fc 20 04 c2 68 85 01   5a
c150 : 4c 31 ea a9 05 85 02 a5   f4
c158 : cb c9 04 f0 0f c9 05 f0   5c
c160 : 31 c9 06 f0 53 c9 03 f0   87
c168 : 75 4c 31 ea 78 a5 01 48   f6
c170 : a9 36 85 01 a2 08 20 ba   16
c178 : ff a9 08 a2 17 a0 c2 20   64
c180 : bd ff a9 b3 a2 00 a0 a8   1c
c188 : 20 d8 ff 68 85 01 58 4c   7c
c190 : 31 ea 78 a5 01 48 a9 36   6f
c198 : 85 01 a2 08 20 ba ff a9   73
c1a0 : 08 a2 1f a0 c2 20 bd ff   f9
c1a8 : a9 b3 a2 a7 a0 af 20 d8   82
c1b0 : ff 68 85 01 58 4c 31 ea   e7
c1b8 : 78 a5 01 48 a9 36 85 01   b1
c1c0 : a2 08 20 ba ff a9 08 a2   78
c1c8 : 27 a0 c2 20 bd ff a9 b3   de
c1d0 : a2 7b a0 b7 20 d8 ff 68   e8
c1d8 : 85 01 58 4c 31 ea 78 a5   15
c1e0 : 01 48 a9 36 85 01 a2 08   31
c1e8 : 20 ba ff a9 08 a2 2f a0   2e
c1f0 : c2 20 bd ff a9 b3 a2 ff   f4
c1f8 : a0 bf 20 d8 ff 68 85 01   f6
c200 : 58 4c 31 ea a2 04 a0 00   f5
c208 : b1 fb 91 fd c8 d0 f9 e6   a4
c210 : fc e6 fe ca d0 f2 60 53   65
c218 : 43 52 45 45 4e 20 31 53   d0
c220 : 43 52 45 45 4e 2d 32 53   44
c228 : 43 52 45 45 4e 2d 33 53   50
c230 : 43 52 45 45 4e 2d 34 ff   b5
```

Listing 1. »SCREENCOPY«, ein Programm zur Verwaltung von bis zu 4 Bildschirmseiten. Bitte beachten Sie die Eingabehinweise auf Seite 7.

```
programm : page4.obj        033c 03ee
-------------------------------------

033c : 20 f1 b7 e0 01 f0 0f e0   f4
0344 : 02 f0 26 e0 03 f0 3d e0   d2
034c : 04 f0 54 4c 48 b2 a5 01   1a
0354 : 48 a9 36 85 01 20 be 03   c1
035c : 85 fb a0 a0 84 fc 20 db   83
0364 : 03 20 c7 03 a0 a4 4c d0   cc
036c : 03 a5 01 48 a9 36 85 01   f0
0374 : 20 be 03 85 fb a0 a8 84   d5
037c : fc 20 db 03 20 c7 03 a0   6d
0384 : ac 4c d0 03 a5 01 48 a9   c2
038c : 36 85 01 20 be 03 85 fb   db
0394 : a0 b0 84 fc 20 db 03 20   7a
039c : c7 03 a0 b4 4c d0 03 a5   46
03a4 : 01 48 a9 36 85 01 20 be   59
03ac : 03 85 fb a0 b8 84 fc 20   69
03b4 : db 03 20 c7 03 a0 bc 4c   d3
03bc : d0 03 a9 00 a0 04 85 fd   b4
03c4 : 84 fe 60 a9 00 a0 d8 85   88
03cc : fd 84 fe 60 85 fb 84 fc   1b
03d4 : 20 db 03 68 85 01 60 a2   d7
03dc : 04 a0 00 b1 fb 91 fd c8   3c
03e4 : d0 f9 e6 fc e6 fe ca d0   3d
03ec : f2 60 00 00 00 00 00 00   0f
```

Listing 2. »PAGE4« zum Einbauen in Ihre eigenen Programme

```
0 REM   ***  SCR.RENAME  ***                   <051>
1 REM   DAZU ZEILE 65 ABAENDERN !!             <246>
2 REM   NAME1 = N1  *  NAME2 = N2
        NAME3 = N3  *  NAME4 = N4              <244>
3 REM               *                          <146>
4 :                                            <236>
5 :N1=49686 : N2=N1+8                          <035>
10 N3=N2+8  : N4=N3+8                          <132>
15 :                                           <247>
20 PRINT CHR$(147) :PRINT                      <011>
25 PRINT" SCREEN-NAME (MAX.8 STELLEN):"        <157>
30 PRINT :PRINT " ";                           <087>
35 :                                           <011>
40 INPUT N$ :N=LEN(N$)                         <036>
45 IF N>8 THEN 20                              <033>
50 IF N<8 THEN N$= N$+CHR$(32) :N=N+1    :
    GOTO 50                                    <002>
55 :                                           <031>
60 FOR I=1 TO N                                <045>
65 :  POKE N1+I,ASC(MID$(N$,I,1))              <150>
70 NEXT                                        <080>
75 END                                         <077>
```

© 64'er

Listing 3. »SCR.RENAME« zum Umbenennen der Bildschirme vor dem Speichern

```
10 sys 9*4096           ;profi-ass aufruf
12 .opt oo
13 *= $c000             ;startadresse
14 ;
20 flag  = $028d
21 taste = $cb
22 irq   = $ea31
29 ;-------------
30 ;
110 sei                     ;init
111 lda #<beg :ldy #>beg
112 sta $0314 :sty $0315
113 cli
115 lda #$00 :ldy #$a0     ;$a000 als
116 sta $b3  :sty $b4      ;zeiger f.save
117 lda #5   :sta $02      ;counter
122 rts
123 ;--------------------
130 ;
131 beg lda flag        ;shift/ctrl-flag
132 :    lsr :bcs tshift
133 :    lsr
134 :    lsr :bcs tctrl
135 :   lda taste
140 :   cmp #4 :beq store1 ;ftaste gedr.
145 :   cmp #5 :beq store2
150 :   cmp #6 :beq store3
155 :   cmp #3 :beq store4
160 :   jmp irq
163 ;---------------------
164 ;
170 tshift jmp shift  ;in akt.bildsch.
171 tctrl dec $02     ;counter =0 "?"
172 lda $02 :beq contr;dann save
190 jmp irq           ;sonst weiter irq
200 contr jmp ctrl
220 ;
221 ;=================================
222 ;store1-4 => der aktuelle bildschirm
  wird im ram (a000-bfff) abgelegt .
225 ;=================================
227 ;
230 store1 jsr up1  :sta $fd
232 :      ldy #$a0 :sty $fe
240 :      jsr copy
245 :      jsr up2  :ldy #$a4
255 :      jmp cop
269 ;
270 store2 jsr up1  :sta $fd
275 :      ldy #$a8 :sty $fe
285 :      jsr copy
290 :      jsr up2  :ldy #$ac
300 :      jmp cop
314 ;
315 store3 jsr up1  :sta $fd
320 :      ldy #$b0 :sty $fe
330 :      jsr copy
335 :      jsr up2  :ldy #$b4
345 :      jmp cop
359 ;
360 store4 jsr up1  :sta $fd
365 :      ldy #$b8 :sty $fe
375 :      jsr copy
380 :      jsr up2  :ldy #$bc
390 :      jmp cop
410 ;-------------------
450 up1 lda #$00 :ldy #$04 ;$0400 nach
455 :   sta $fb  :sty $fc  ;$fb/c
460 :   rts
464 ;
465 up2 lda #$00 :ldy #$d8 ;$d800 nach
470 :   sta $fb  :sty $fc  ;$fb/c
475 :   rts
```

Listing 4. Source-Code-Listing zu »SCREENCOPY«

```
479 ;
480 cop sta $fd :sty $fe
485 :    jsr copy
490 :    jmp irq
491 ;---------------------
492 shift lda taste
493 :      cmp #4 :beq hole1
494 :      cmp #5 :beq hole2
495 :      cmp #6 :beq hole3
496 :      cmp #3 :beq hole4
497 :      jmp irq
498 ;---------------------
499 ;
500 ;==================================
501 ;hole1-4 => der gespeicherte bildsch
irm wird in d.aktuellen zurueckkopiert.
502 ;==================================
505 hole1 lda $01   :pha
510 :      lda #$36 :sta $01
515 :      jsr up3  :sta $fb
520 :      ldy #$a0 :sty $fc
525 :      jsr copy
530 :      jsr up4  :ldy #$a4
540 :      jmp cop2
545 ;
555 hole2 lda $01   :pha
560 :      lda #$36 :sta $01
565 :      jsr up3  :sta $fb
570 :      ldy #$a8 :sty $fc
580 :      jsr copy
585 :      jsr up4  :ldy #$ac
595 :      jmp cop2
600 ;
610 hole3 lda $01   :pha
615 :      lda #$36 :sta $01
620 :      jsr up3  :sta $fb
625 :      ldy #$b0 :sty $fc
635 :      jsr copy
640 :      jsr up4  :ldy #$b4
650 :      jmp cop2
655 ;
665 hole4 lda $01   :pha
670 :      lda #$36 :sta $01
675 :      jsr up3  :sta $fb
680 :      ldy #$b8 :sty $fc
690 :      jsr copy
695 :      jsr up4  :ldy #$bc
705 :      jmp cop2
709 ;---------------------
710 ;
720 up3 lda #$00 :ldy #$04 ;$0400 nach
725 :    sta $fd  :sty $fe  ;$fd/e
730 :    rts
734 ;
735 up4 lda #$00 :ldy #$d8 ;$d800 nach
740 :    sta $fd  :sty $fe  ;$fd/e
745 :    rts
749 ;
750 cop2 sta $fb :sty $fc
755 :    jsr copy
765 :    pla :sta $01
770 :    jmp irq
775 ;---------------------
776 ctrl lda #5 :sta $02 ;setze zaehler
780 :    lda taste        ;ctrl/f-taste
781 :    cmp #4 :beq save1;gedrueckt"?"
782 :    cmp #5 :beq save2
783 :    cmp #6 :beq save3
784 :    cmp #3 :beq save4
785 :    jmp irq
786 ;
788 ;==================================
789 ;save 1-4 =>die bildschirme unter
  dem rom werden abgespeichert.
790 ;==================================
```

```
791 save1 sei
792 :     lda $01    :pha
794 :     lda #$36   :sta $01
795 :     ldx #8     :jsr $ffba
796 :     lda #8
797 :     ldx #<na1 :ldy #>na1
798 :     jsr $ffbd
799 :     lda #$b3
800 :     ldx #$00   :ldy #$a8
801 :     jsr $ffd8
802 :     pla :sta $01
803 :     cli :jmp irq
805 ;
810 save2 sei
811 :     lda $01    :pha
812 :     lda #$36   :sta $01
813 :     ldx #8     :jsr $ffba
814 :     lda #8
815 :     ldx #<na2 :ldy #>na2
816 :     jsr $ffbd
817 :     lda #$b3
818 :     ldx #$a7   :ldy #$af
819 :     jsr $ffd8
820 :     pla :sta $01
822 :     cli :jmp irq
825 ;
830 save3 sei
831 :     lda $01    :pha
832 :     lda #$36   :sta $01
833 :     ldx #8     :jsr $ffba
834 :     lda #8
835 :     ldx #<na3 :ldy #>na3
836 :     jsr $ffbd
837 :     lda #$b3
838 :     ldx #$7b   :ldy #$b7
839 :     jsr $ffd8
840 :     pla :sta $01
843 :     cli :jmp irq
845 ;
850 save4 sei
851 :     lda $01    :pha
852 :     lda #$36   :sta $01
853 :     ldx #8     :jsr $ffba
854 :     lda #8
855 :     ldx #<na4 :ldy #>na4
856 :     jsr $ffbd
857 :     lda #$b3
858 :     ldx #$ff   :ldy #$bf
859 :     jsr $ffd8
860 :     pla :sta $01
862 :     cli :jmp irq
900 ;====================================
901 ;startadresse ==> in $fb/c
  endadresse   ==> in $fd/e
902 ;die angegebenen bereiche werden mit
  diesem unterprg. kopiert.
903 ;====================================
905 copy ldx #4 :ldy #0
910 n lda ($fb),y
915 : sta ($fd),y
920 : iny
925 : bne n
930 : inc $fc :inc $fe
935 : dex
940 : bne n
945 : rts
950 ;---------------------
960 na1 .asc "screen 1"
961 na2 .asc "screen-2"
962 na3 .asc "screen-3"
963 na4 .asc "screen-4"
READY.
```

Listing 4. »SCREENCOPY« (Schluß)

```
10 sys 9*4096        ;profi-ass
12 .opt oo
13 *= $033c          ;startadresse
14 ;
50 illq   = $b248
55 koget = $b7f1
90 ;
100 jsr koget
110 cpx #1 :beq hole1
120 cpx #2 :beq hole2
130 cpx #3 :beq hole3
140 cpx #4 :beq hole4
150 jmp illq
200 ;
201 ;==================================
202 ;die gespeich.screens werden ueber
 sys-befehl->sys828,n (n=1-4)aufgerufen
203 ;==================================
500 hole1 lda $01   :pha
510 :       lda #$36 :sta $01
515 :       jsr up3  :sta $fb
520 :       ldy #$a0 :sty $fc
525 :       jsr copy
530 :       jsr up4  :ldy #$a4
540 :       jmp cop2
545 ;
555 hole2 lda $01   :pha
560 :       lda #$36 :sta $01
565 :       jsr up3  :sta $fb
570 :       ldy #$a8 :sty $fc
580 :       jsr copy
585 :       jsr up4  :ldy #$ac
595 :       jmp cop2
600 ;
610 hole3 lda $01   :pha
615 :       lda #$36 :sta $01
620 :       jsr up3  :sta $fb
625 :       ldy #$b0 :sty $fc
635 :       jsr copy
640 :       jsr up4  :ldy #$b4
650 :       jmp cop2
655 ;
665 hole4 lda $01   :pha
670 :       lda #$36 :sta $01
675 :       jsr up3  :sta $fb
680 :       ldy #$b8 :sty $fc
690 :       jsr copy
695 :       jsr up4  :ldy #$bc
705 :       jmp cop2
710 ;
720 up3 lda #$00 :ldy #$04 ;$0400 nach
725 :    sta $fd  :sty $fe  ;$fd/e
730 :    rts
734 ;
735 up4 lda #$00 :ldy #$d8 ;$d800 nach
740 :    sta $fd  :sty $fe  ;$fd/e
745 :    rts
749 ;
750 cop2 sta $fb :sty $fc
755 :     jsr copy
765 :     pla :sta $01
770 :     rts                ;fertig
775 ;
900 copy ldx #4 :ldy #0   ;kopierroutine
905 n lda ($fb),y
910 : sta ($fd),y
915 : iny
920 : bne n
925 : inc $fc :inc $fe
930 : dex
935 : bne n
940 rts
```

Listing 5. Source-Code-Listing zu »PAGE4«

64ER ONLINE

Joystickabfrage in den Interrupt verlegt

Beeinflussen Sie mit dem Joystick ein Sprite während des Programmablaufs. Ein optimaler Zusatz um Spiele zu programmieren und Menüs zu unterstützen.

Ab jetzt haben Sie mit der Abfrage Ihres Joystick keine Probleme mehr. Dieses Programm übernimmt das für Sie vollständig und bewegt entsprechend der Joystickbewegungen das Sprite mit der Nummer 1. Das Programm wurde in Maschinensprache geschrieben, um eine möglichst hohe Geschwindigkeit zu erreichen. Um das Programm vom Basic-Ablauf zu trennen, wurde es in den Interrupt eingebaut. Gesteuert wird dabei mit dem Joystick in Port 2. Das Programm (Listing 1) selbst befindet sich im Kassettenpuffer und wird mit »SYS 951« gestartet. Das entsprechende Sprite (Nummer 1) läßt sich auf den gesamten Bildschirm bewegen (auch über den kritischen Punkt, bei dem das Low-Byte der X-Achse den Wert 255 übersteigt) und läuft nicht über den Rand hinaus. Versuchen Sie doch einmal das kleine Demo-Programm (Listing 2).

Um das Programm auf Port 1 umzustellen, ändern Sie mittels POKE in den folgenden Speicherstellen den Wert von »0« auf »1«: 829, 839, 849 und 854. Der Knopf des Joysticks wird vom Programm nicht berücksichtigt.

(H. Werner/og)

```
PROGRAMM : JOY IRQ              033C 03C4
-----------------------------------------

033C : AD 00 DC 29 01 D0 03 CE  86
0344 : 01 D0 AD 00 DC 29 02 D0  D9
034C : 03 EE 01 D0 AD 00 DC 29  C1
0354 : 04 D0 03 CE 00 D0 AD 00  98
035C : DC 29 08 D0 03 EE 00 D0  32
0364 : AD 10 D0 C9 01 D0 1A AD  E1
036C : 00 D0 C9 40 D0 03 CE 00  AF
0374 : D0 AD 00 D0 C9 00 D0 08  25
037C : CE 00 D0 A9 00 8D 10 D0  02
0384 : 4C A0 03 AD 00 D0 C9 16  71
038C : D0 03 EE 00 D0 AD 00 D0  B6
0394 : C9 FF D0 08 A9 01 8D 00  6B
039C : D0 8D 10 D0 AD 01 D0 C9  0B
03A4 : E6 D0 03 CE 01 D0 AD 01  DC
03AC : D0 C9 31 D0 03 EE 01 D0  15
03B4 : 4C 31 EA 78 A9 3C 8D 14  3D
03BC : 03 A9 03 8D 15 03 58 60  92
```

Listing 1. »JOY IRQ« geben Sie bitte mit dem MSE (Seite 7) ein

```
10 POKE 2053,143:LOAD"JOY IRQ",8,1          <042>
20 SYS 951        :REM INITIALISIERUNG      <191>
30 POKE V+21,1    :REM SPRITE EIN           <046>
40 POKE V+39,1    :REM FARBE WEISS          <038>
50 POKE V+0,160   :REM SPRITE IN...         <173>
60 POKE V+1,120   :REM ...DIE MITTE         <123>
70 POKE 2040,11   :REM SPRITEZEIGER...      <247>
75 :               REM ...AUF BLOCK 11      <196>
80 FOR X=704 TO 766:POKE X,255:NEXT         <020>
85 :      REM SPRITEINHALT DEFINIEREN       <039>
```

Listing 2. Ein kleines Demo-Programm für »JOY IRQ«

Und er LISTet doch!

Man kann jedes Basic-Programm LISTen, auch wenn es mit einem LIST-Schutz versehen ist. Das Programm »LISTKNACKER« erledigt das für Sie – eventuelle POKEs oder Steuerzeichen hindern das Programm nicht am LISTen.

Stellen Sie sich vor, Sie haben ein Programm mit einem (fast) perfekten LIST-Schutz versehen, und wollen nach geraumer Zeit eine Verbesserung vornehmen. Doch die Version ohne Schutz ist nicht aufzufinden, und wie Sie das Programm damals geschützt haben, wissen Sie auch nicht mehr so genau. Was tun, um an ein Listing zu kommen? Ganz einfach: Sie nehmen den »LISTKNACKER« und holen sich Ihr Listing direkt von der Diskette.

Vom »LISTKNACKER« gibt es zwei Versionen: eine, um auf dem Bildschirm zu LISTen (Listing 1), und die andere, um das Listing auf den Drucker auszugeben (Listing 2). Geben Sie also die gewünschte Version mit dem Checksummer ein, speichern Sie sie und starten sie mit RUN. Dann legen Sie die Diskette mit dem zu listenden Programm ein und bestätigen dies durch Betätigen der CONTROL-Taste. Daraufhin geben Sie den Namen des Programms ein. Auf dem Bildschirm erscheinen die ASCII-Werte der Steuerzeichen mit vorangestelltem Komma in weiß (zum Beispiel »cursor down« als »,17«), die restlichen Zeichen und Befehle in hellblau. Auf einem Drucker sind die entsprechenden Zeichen revers dargestellt. Als Drucker lassen sich alle diejenigen verwenden, bei denen »OPEN 4,4:CMD4:LIST« zum korrekten LISTen des Programms führt.

(G. Engist/og)

Zeile 10 Sprung nach Zeile 220

Zeile 220–360
START

320	Programmfile öffnen
330–340	Fehlermeldung von Floppy
350	in indizierte Variable A$( ) werden Basic-Befehle eingelesen
360	2 Byte, die die Startadresse ergeben, werden überlesen

↓

Zeile 160–210
ZEILENNUMMER berechnen und ausgeben, über Programmende entscheiden die Zeilen

160	die 2 Byte, die auf die nächste Zeile zeigen, werden überlesen
180	Statusvariable = 66 entspricht Programmfile-Ende
210	A=0 entspricht Gänsefüßchenmodus aus

↓ ↑

Zeile 20–150
KERN

30	A$=" " entspricht Zeiger auf nächste Zeilennummer
50–110	entscheiden über Gänsefüßchenmodus (A=0 aus, A=1 ein)
120–130	nachprüfen, ob Basic-Befehl oder nicht; wenn ja, wird die indizierte Variable A$( ) benützt

64er-online.de

```
7 REM       VON                            <180>
8 REM  GEORG ENGIST                        <251>
9 REM 7818 ACHKARREN                       <072>
10 GOTO 220                                <234>
20 POKE 646,14:GET#2,A$                    <032>
30 IF A$=""GOTO 160                        <203>
40 B=ASC(A$)                               <194>
50 IF B=G THEN IF A THEN A=H:GOTO 120      <180>
60 IF B=G THEN A=I                         <092>
70 IF A=H GOTO 120                         <062>
80 IF B>J THEN IF B<K THEN PRINT A$;:GOTO
   20                                      <178>
90 IF B>L THEN PRINT A$;:GOTO 20           <035>
100 B$=","+RIGHT$(STR$(B),LEN(STR$(B))-1)  <152>
110 POKE 646,I:PRINT B$;:GOTO 20           <113>
120 IF B>M THEN IF B<N THEN PRINT A$(B-P);
   :GOTO 20                                <239>
130 IF B>N THEN IF B<Q THEN PRINT A$(B-R);
   :GOTO 20                                <045>
140 IF B>J THEN PRINT A$;:GOTO 20          <021>
150 PRINT CHR$(5)","RIGHT$(STR$(B),LEN(STR
   $(B))-1);:GOTO 20                       <045>
160 FOR Z=1 TO 3:GET#2,A$:NEXT Z           <107>
170 GET#2,B$:A$=A$+CHR$(.):B$=B$+CHR$(.)   <231>
180 IF ST=66 THEN CLOSE 1:CLOSE 2:END      <090>
190 C=ASC(A$):D=ASC(B$):E=256*D+C          <034>
200 A$=STR$(E):B$=RIGHT$(A$,LEN(A$)-1)+" " <021>
210 PRINT:PRINT B$;:A=.:GOTO 20            <255>
220 CLR:PRINT CHR$(142)CHR$(147)CHR$(5)"{3
   DOWN}                                   <208>
230 PRINT"{2SPACE}R{2SPACE}R UI ARS SUI UI
   R UI UI SUI AS AI                       <131>
240 PRINT"{2SPACE}-{2SPACE}- JI{2SPACE}-{2
   SPACE}QEI -JW QW -{2SPACE}QEI Q{2SPACE
   }QW                                     <081>
250 PRINT"{2SPACE}ZX E{SHIFT-SPACE}JK{2SPA
   CE}E{2SPACE}X J E E KJ JK X J ZX EJ{3D
   OWN}                                    <224>
260 PRINT TAB(17)"V O N                    <010>
270 PRINT TAB(9)"G E O R G{2SPACE}E N G I
   S T"CHR$(154)                           <103>
280 G=34:I=1:J=31:K=128:L=159:M=127:N=204:
   P=128:Q=256:R=205                       <255>
290 PRINT TAB(7)"{3DOWN}DISKETTE EINLEGEN
   ! (CTRL){DOWN}":POKE 649,.:WAIT 653,4:
   POKE 649,10                             <033>
300 OPEN 1,8,15,"I0":PRINT TAB(4)          <080>
310 INPUT"PROGRAMMNAME ";NA$               <181>
320 OPEN 2,8,0,"0:"+NA$                    <012>
330 INPUT#1,EN,EN$                         <107>
340 IF EN THEN PRINT TAB(17)"{2DOWN}"EN:PR
   INT TAB(20-INT(LEN(EN$)/2))EN$:END      <041>
350 DIM A$(75):FOR Z=.TO 75:READ A$(Z):NEX
   T Z:PRINT CHR$(147)                     <143>
360 FOR Z=.TO 1:GET#2,A$:NEXT Z:GOTO 160   <236>
370 DATA END,FOR,NEXT,DATA,"INPUT#"        <011>
380 DATA INPUT,DIM,READ,LET,GOTO,RUN       <014>
390 DATA IF,RESTORE,GOSUB,RETURN           <251>
400 DATA REM,STOP,ON,WAIT,LOAD,SAVE        <107>
410 DATA VERIFY,DEF,POKE,"PRINT#",PRINT    <252>
420 DATA CONT,LIST,CLR,CMD,SYS,OPEN        <054>
430 DATA CLOSE,GET,NEW,"TAB(",TO,FN        <061>
440 DATA"SPC(",THEN,NOT,STEP,+             <219>
450 DATA-,*,/,↑,AND,OR,">"                 <233>
460 DATA=,"<",SGN,INT,ABS,USR,FRE          <217>
470 DATA POS,SQR,RND,LOG,EXP,COS,SIN       <220>
480 DATA TAN,ATN,PEEK,LEN,"STR$",VAL       <193>
490 DATA ASC,"CHR$","LEFT$","RIGHT$"       <010>
500 DATA"MID$",GO                          <070>
```

© 64'er

Listing 1. »LISTKNACKER« LISTet jedes Basic-Programm auf den Bildschirm

```
7 REM       VON                            <180>
8 REM  GEORG ENGIST                        <251>
9 REM 7818 ACHKARREN                       <072>
10 GOTO 220                                <234>
20 GET#2,A$                                <189>
30 IF A$=""GOTO 160                        <203>
40 B=ASC(A$)                               <194>
50 IF B=G THEN IF A THEN A=H:GOTO 120      <180>
60 IF B=G THEN A=I                         <092>
70 IF A=H GOTO 120                         <062>
80 IF B>J THEN IF B<K THEN PRINT#4,A$;:GOT
   O 20                                    <102>
90 IF B>L THEN PRINT#4,A$;:GOTO 20         <202>
100 B$=CHR$(18)+","+RIGHT$(STR$(B),LEN(STR
   $(B))-1)+CHR$(146)                      <172>
110 PRINT#4,B$;:GOTO 20                    <183>
120 IF B>M THEN IF B<N THEN PRINT#4,A$(B-P
   );:GOTO 20                              <159>
130 IF B>N THEN IF B<Q THEN PRINT#4,A$(B-R
   );:GOTO 20                              <245>
140 IF B>J THEN PRINT#4,A$;:GOTO 20        <188>
150 PRINT#4,CHR$(18);",";RIGHT$(STR$(B),LE
   N(STR$(B))-1);CHR$(146):GOTO 20         <015>
160 FOR Z=1 TO 3:GET#2,A$:NEXT Z           <107>
170 GET#2,B$:A$=A$+CHR$(.):B$=B$+CHR$(.)   <231>
180 IF ST=66 THEN CLOSE 1:CLOSE 2:CLOSE 4:
   END                                     <020>
190 C=ASC(A$):D=ASC(B$):E=256*D+C          <034>
200 A$=STR$(E):B$=RIGHT$(A$,LEN(A$)-1)+" " <021>
210 PRINT#4:PRINT#4,B$;:A=.:GOTO 20        <180>
220 CLR:PRINT CHR$(142)CHR$(147)CHR$(5)"{3
   DOWN}                                   <208>
230 PRINT"{2SPACE}R{2SPACE}R UI ARS SUI UI
   R UI UI SUI AS AI                       <131>
240 PRINT"{2SPACE}-{2SPACE}- JI{2SPACE}-{2
   SPACE}QEI -JW QW -{2SPACE}QEI Q{2SPACE
   }QW                                     <081>
250 PRINT"{2SPACE}ZX E{SHIFT-SPACE}JK{2SPA
   CE}E{2SPACE}X J E E KJ JK X J ZX EJ{3D
   OWN}                                    <224>
260 PRINT TAB(17)"V O N                    <010>
270 PRINT TAB(9)"G E O R G{2SPACE}E N G I
   S T"CHR$(154)                           <103>
280 G=34:I=1:J=31:K=128:L=159:M=127:N=204:
   P=128:Q=256:R=205                       <255>
290 PRINT TAB(7)"{3DOWN}DISKETTE EINLEGEN
   ! (CTRL){DOWN}":POKE 649,.:WAIT 653,4:
   POKE 649,10                             <033>
300 OPEN 1,8,15,"I0":PRINT TAB(4)          <080>
310 INPUT"PROGRAMMNAME ";NA$               <181>
320 OPEN 2,8,0,"0:"+NA$                    <012>
330 INPUT#1,EN,EN$                         <107>
340 IF EN THEN PRINT TAB(17)"{2DOWN}"EN:PR
   INT TAB(20-INT(LEN(EN$)/2))EN$:END      <041>
350 DIM A$(75):FOR Z=.TO 75:READ A$(Z):NEX
   T Z:PRINT CHR$(147):OPEN 4,4            <004>
360 FOR Z=.TO 1:GET#2,A$:NEXT Z:GOTO 160   <236>
370 DATA END,FOR,NEXT,DATA,"INPUT#"        <011>
380 DATA INPUT,DIM,READ,LET,GOTO,RUN       <014>
390 DATA IF,RESTORE,GOSUB,RETURN           <251>
400 DATA REM,STOP,ON,WAIT,LOAD,SAVE        <107>
410 DATA VERIFY,DEF,POKE,"PRINT#",PRINT    <252>
420 DATA CONT,LIST,CLR,CMD,SYS,OPEN        <054>
430 DATA CLOSE,GET,NEW,"TAB(",TO,FN        <061>
440 DATA"SPC(",THEN,NOT,STEP,+             <219>
450 DATA-,*,/,↑,AND,OR,">"                 <233>
460 DATA=,"<",SGN,INT,ABS,USR,FRE          <217>
470 DATA POS,SQR,RND,LOG,EXP,COS,SIN       <220>
480 DATA TAN,ATN,PEEK,LEN,"STR$",VAL       <193>
490 DATA ASC,"CHR$","LEFT$","RIGHT$"       <010>
500 DATA"MID$",GO                          <070>
```

© 64'er

Listing 2. »LISTKNACKER« für den Drucker

Ordnung ist das halbe Leben

Mit diesem Programm bringen Sie auf bequeme Weise Ordnung in die Directories Ihrer Disketten. Nicht nur eine verbesserte Optik des Inhaltsverzeichnisses hilft Ihnen bei der Suche nach bestimmten Programmen.

Ist Ihnen das auch schon mehrmals passiert: Sie hatten auf einer Diskette ein schön geordnetes Directory und beim Speichern eines weiteren Files stand dieses nicht sauber am Schluß des Directory, sondern mitten zwischen den anderen Programmen. Mit dem hier vorgestellten Directory-Sortierprogramm ist es nun möglich, diese »falsch hineingeratenen« Files herauszunehmen und an geeigneter Stelle wieder einzusetzen. Auch das Einfügen von Trennstrichen ist möglich. Bild 1 und Bild 2 zeigen Ihnen den Unterschied eines unsortierten zu einem sortierten Directory. Mit etwas Geschick kann man seine Disketten auch so umsortieren, daß an erster Stelle immer ein Ladeprogramm für das Hauptprogramm steht. Man kann sich dadurch viel Sucharbeit sparen, denn man lädt einfach das erste Programm.

Nach dem Starten des Programms legt man die Diskette ein, die sortiert werden soll und drückt eine Taste. In der linken oberen Ecke wird nun die Sektornummer des Directory-Blocks angezeigt, den der Computer gerade einliest, rechts daneben die Anzahl der in den Speicher eingelesenen Files.

Direktory mit System

Nach beendetem Einlesen erscheint links vom obersten File ein schwarzer Pfeil, der sich mit den Funktionstasten F3 nach oben und F5 nach unten bewegen läßt. Mittels F1 kann nun ein mit diesem Pfeil gekennzeichnetes File nach rechts herausgeschoben und mit F3 und F5 verschoben werden. An gewünschter Stelle wird es mit F1 wieder eingefügt. Somit ist ein beliebiges Vertauschen aller Files im Directory möglich.

F8 schreibt das sortierte Directory wieder auf die Diskette zurück. Will man das geänderte Directory nicht gespeichert haben, so kann statt dessen mit F6 noch einmal das alte oder ein anderes eingelesen werden.

Zur optischen Abgrenzung mehrerer Files dient der Trennstrich, den man mit F2 erzeugen, mit F3 und F5 verschieben und schließlich wieder mit F1 einfügen kann. Wem übrigens ein anderer Trennstrich besser gefällt, der kann in Zeile 260 für die Minuszeichen andere einsetzen, zum Beispiel SHIFT+ *. SHIFT + C ist nicht zu empfehlen, da dieser Strich bei Groß-Kleinschrift-Umschaltung ein großes C ergibt. Der Trennstrich belegt keinen Block auf der Diskette und ist zur besseren Unterscheidung mit DEL im Directory gekennzeichnet. Ein versehentlich mit »F2« erzeugter Trennstrich kann, wenn er nach rechts gebracht wird, mit F4 wieder gelöscht werden. In gleicher Weise können auch Files aus dem Directory gelöscht werden. Dabei werden aber die von dem File belegten Blöcke nicht wieder freigegeben, so daß ein abschließendes »VALIDATE« der Diskette erforderlich ist, wenn man ein File gelöscht hat.

Nun zum Aufbau des Programms. Es ist vorwiegend in Basic geschrieben und enthält zwei kurze Maschinensprache-Routinen. Die erste dient zum Einlesen der Filenamen und Fileparameter und wird mit SYS 52992, log. Filenummer, Länge, String aufgerufen. Das Lesen mit GET # in Basic wäre dafür viel zu langsam. Mit dem zweiten Maschinenprogramm werden die am Bildschirm angezeigten Files nach oben oder unten gescrollt. Aufgerufen wird es mit SYS 53056,r,oz,ls,uz,rs.

Dabei haben die Parameter folgende Bedeutung:

r = Scrollrichtungen: 0 für nach oben und 1 für nach unten
oz = oberste Zeile
ls = linke Spalte
uz = unterste Zeile
rs = rechte Spalte

Durch Speichern des Speicherbereichs von $CF00 bis $CFFF kann diese Routine auch in andere Basic- oder Maschinenprogramme eingebunden werden.

(Edwin Göbel/ah)

```
0 "64'ER 11/85      " 64'ER
16   "1. PRG"              PRG
7    "MSE V1.0"            PRG
58   "APFELMANN"           PRG
13   "HARDCOPY KOALA"      PRG
12   "2. PRG"              PRG
69   "PROFIPRINT"          PRG
4    "APFELROUTINEN"       PRG
69   "LYRIC 3.0"           PRG
12   "CHECKSUMMER V3"      PRG
32   "APFELMANN.PIC"       PRG
41   "0. PRG"              PRG
26 BLOCKS FREE.

READY.
```

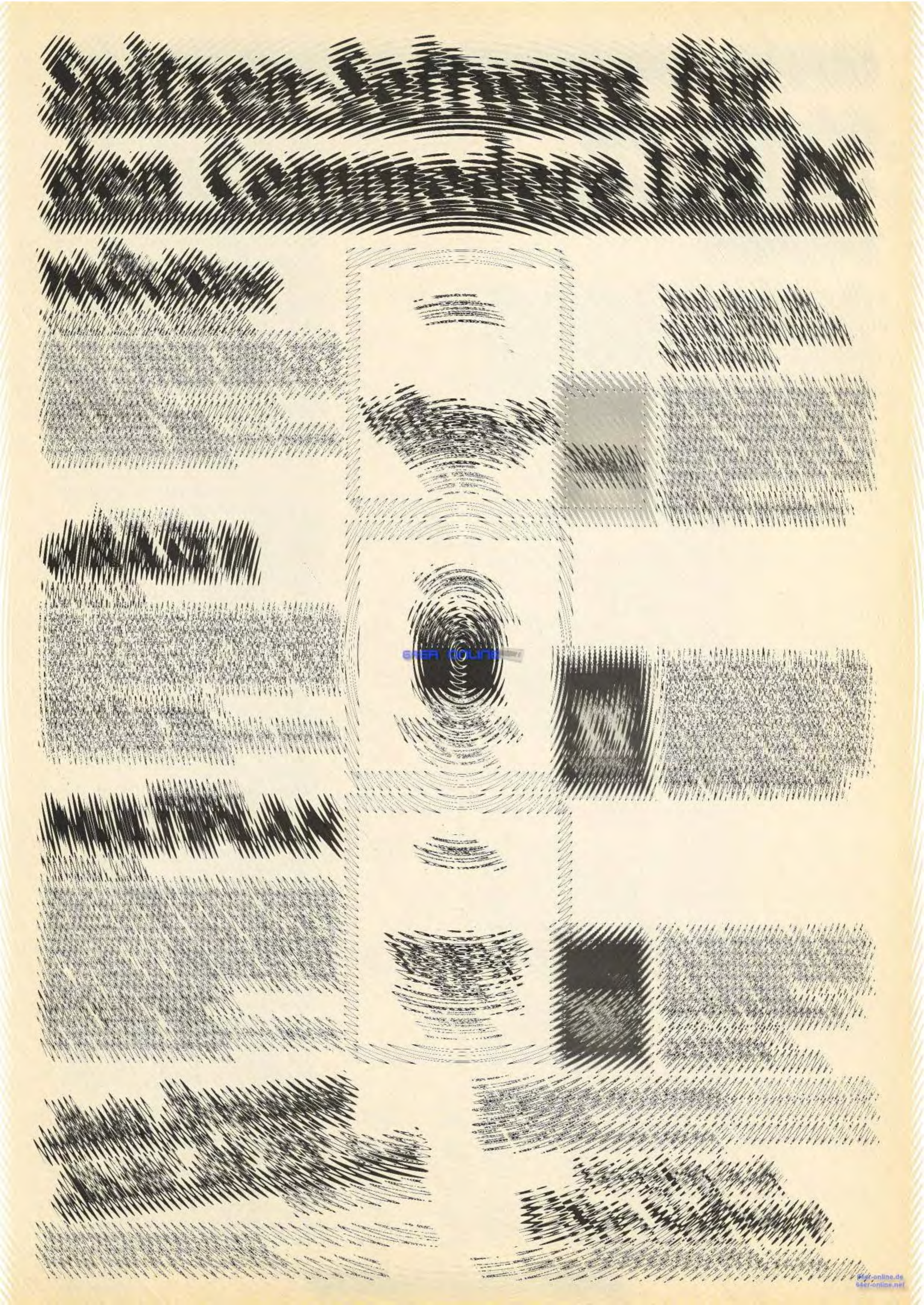
Bild 1. Das Durcheinander eines unsortierten Directorys

```
0 "64'ER 11/85      " 64'ER
12   "CHECKSUMMER V3"      PRG
7    "MSE V1.0"            PRG
0    "----------------"    DEL
13   "HARDCOPY KOALA"      PRG
0    "----------------"    DEL
69   "LYRIC 3.0"           PRG
0    "----------------"    DEL
41   "0. PRG"              PRG
16   "1. PRG"              PRG
12   "2. PRG"              PRG
0    "  HYPRA PLATOS  "    PRG
0    "----------------"    DEL
69   "PROFIPRINT"          PRG
0    "----------------"    DEL
4    "APFELROUTINEN"       PRG
58   "APFELMANN"           PRG
32   "APFELMANN.PIC"       PRG
26 BLOCKS FREE.
```

Bild 2. Der »Directory-Sorter« schafft schnell Ordnung

```
10 GOTO 100                                      <194>
20 INPUT#1,F1,F$,F2,F3:IF F1=0 THEN RETURN       <049>
30 PRINT F1;F$,F2;F3:END                         <196>
40 IF NO+EN>=C THEN RETURN                       <209>
50 SYS 53056,0,2,2,23,18:NO=NO+1                 <182>
60 POKE 214,23:POKE 211,2:SYS 58732:PRINT
   NA$(ZU(NO+EN-1));:RETURN                      <238>
70 IF NO=0 THEN RETURN                           <157>
80 SYS 53056,1,2,2,23,18:NO=NO-1                 <218>
90 POKE 214,2:POKE 211,2:SYS 58732:PRINT N
   A$(ZU(NO+1)):RETURN                           <195>
100 POKE 53280,6:POKE 53281,6                    <008>
110 PRINT"{CLR,YELLOW,RVSON}"TAB(9)"EPLOX'
    S DIRECTORY-SORT"                            <034>
120 PRINT TAB(8)"------------------------"
    :IF DG=1 THEN 200                            <175>
130 PRINT" F1{2SPACE}EINTRAG MARKIEREN/EIN
    ORDNEN"                                      <192>
140 PRINT" F2{2SPACE}TRENNSTRICH ERZEUGEN"       <017>
150 PRINT" F3{2SPACE}AUF"                        <091>
160 PRINT" F4{2SPACE}LOESCHEN"                   <154>
170 PRINT" F5{2SPACE}AB"                         <190>
180 PRINT" F8{2SPACE}SPEICHERN"                  <250>
190 PRINT "{3SPACE,3DOWN}BITTE DISKETTE EI
    NLEGEN":DG=1:POKE 198,0:WAIT 198,63:GO
    TO 110                                       <240>
200 OPEN 1,8,15,"I":DIM AN$(145),NA$(145),
    RE$(145),ZU(146),SN(19)                      <234>
210 FOR I=1 TO 18:READ SN(I):NEXT                <166>
220 FOR I=52992 TO 53242:READ X:POKE I,X:N
    EXT                                          <047>
230 GOSUB 20:OPEN 2,8,2,"#":GOSUB 20             <081>
240 S=1:C=1:N$=CHR$(0):NN$=N$+N$+N$              <177>
250 FOR I=1 TO 10:NU$=NU$+NN$:NEXT               <005>
260 AN$(0)=CHR$(128)+CHR$(18)+CHR$(1):NA$(
    0)="----------------"                        <213>
270 RE$(0)=NN$+NN$+NN$+N$+N$                     <049>
280 PRINT#1,"U1 2 0 18"S:PRINT"{HOME,LIG.R
    ED}"S"{LEFT,SPACE}"                          <153>
290 GET#2,T$:GET#2,S$:S=ASC(S$+N$)               <146>
300 FOR BP=0 TO 7:PRINT#1,"B-P 2";BP*32+2        <023>
310 SYS 52992,2,3,X$:AN$(C)=X$:IF LEFT$(X$
    ,1)=N$THEN NEXT:GOTO 340                     <097>
320 SYS 52992,2,16,X$:NA$(C)=X$:SYS 52992,
    2,11,X$:RE$(C)=X$                            <066>
330 ZU(C)=C:PRINT"{HOME,4RIGHT}"C"{LEFT,SP
    ACE}":C=C+1:NEXT                             <236>
340 IF T$<>""THEN 280                            <088>
350 CLOSE 2:POKE 650,128:PRINT"{HOME,2DOWN
    ,BLACK}>{CYAN,UP}";:CP=2:NO=0:EN=C:IF
    EN>23 THEN EN=23                             <239>
360 PRINT CHR$(13)TAB(2)NA$(NU+1);:NU=NU+1
    :IF NU<EN-1 THEN 360                         <128>
370 GET TA$:IF TA$=""THEN 370                    <001>
380 IF TA$="{F3}"THEN IF CP>2 THEN CP=CP-1
    :SYS 53056,0,2,0,23,0                        <007>
390 IF TA$="{F3}"THEN IF CP=2 THEN GOSUB 7
    0                                            <147>
400 IF TA$="{F5}"THEN IF CP<EN THEN CP=CP+
    1:SYS 53056,1,2,0,23,0                       <223>
410 IF TA$="{F5}"THEN IF CP>=23 THEN GOSUB
     40                                          <116>
420 IF TA$="{F1}"THEN 480                        <092>
430 IF TA$="{F2}"THEN IF C<145 THEN F=1:C=
    C+1:PRINT"{LIG.RED,HOME,4RIGHT}"C-1"{L
    EFT,SPACE,CYAN}":EN=EN+1:IF EN>23 THEN
     EN=23                                       <078>
440 IF TA$="{F2}"THEN IF F=1 THEN F=0:TE=0
    :TE$=NA$(0):GOTO 510                         <121>
450 IF TA$="{F8}"THEN 670                        <129>
460 IF TA$="{HOME}"THEN 730                      <218>
470 GOTO 370                                     <016>
480 TE=ZU(NO+CP-1):TE$=NA$(TE)                   <226>
490 SYS 53056,0,CP,2,23,18:IF C>23 THEN EN
    =EN+1:GOSUB 60:EN=EN-1                       <082>
500 FOR I=NO+CP-2 TO C-1:ZU(I+1)=ZU(I+2):N
    EXT                                          <073>
510 POKE 214,CP:POKE 211,19:SYS 58732            <225>
520 PRINT"TT"TE$                                 <112>
530 GET TA$:IF TA$=""THEN 530                    <160>
540 IF TA$="{F3}"THEN IF CP>2 THEN CP=CP-1
    :SYS 53056,0,2,19,24,37:SYS 53056,0,2,
    0,23,0                                       <233>
550 IF TA$="{F3}"THEN IF CP=2 THEN GOSUB 7
    0                                            <053>
560 IF TA$="{F5}"THEN IF CP<EN THEN CP=CP+
    1:SYS 53056,1,2,19,24,37:SYS 53056,1,2
    ,0,23,0                                      <041>
570 IF TA$="{F5}"THEN IF CP=23 THEN GOSUB
    40                                           <135>
580 IF TA$="{F1}"THEN 620                        <174>
590 IF TA$="{F4}"THEN TE$="{16RIGHT}":C=C-
    1:EN=C:IF EN>23 THEN EN=23                   <224>
600 IF TA$="{F4}"THEN PRINT"{HOME,LIG.RED,
    4RIGHT}"C-1"{LEFT,SPACE,CYAN}":GOTO 64
    0                                            <010>
610 GOTO 530                                     <110>
620 SYS 53056,1,CP,2,23,18                       <012>
630 FOR I=C-1 TO NO+CP-1 STEP-1:ZU(I+1)=ZU
    (I):NEXT:ZU(NO+CP-1)=TE                      <209>
640 POKE 214,CP:POKE 211,2:SYS 58732             <057>
650 PRINT TE$"{19SPACE}"                         <007>
660 GOTO 370                                     <208>
670 OPEN 2,8,2,"#":T=18:FOR I=0 TO INT((C-
    2)/8):IF 8*I+8>=C-1 THEN T=0                 <103>
680 PRINT#1,"B-P 2 0":PRINT#2,CHR$(T)CHR$(
    SN(I+2))                                     <121>
690 FOR BP=0 TO 7:PRINT#1,"B-P 2";BP*32+2        <159>
700 IF BP+8*I+1>=C THEN PRINT#2,NU$;:NEXT:
    GOTO 720                                     <023>
710 PRINT#2,AN$(ZU(BP+8*I+1))NA$(ZU(BP+8*I
    +1))RE$(ZU(BP+8*I+1));:NEXT                  <011>
720 PRINT#1,"U2 2 0 18"SN(I+1):PRINT"{LIG.
    RED,HOME}"TAB(36)SN(I+1)"{LEFT,SPACE,C
    YAN}":GOSUB 20:NEXT                          <018>
730 CLOSE 1:CLOSE 2:RUN                          <211>
740 DATA 1,4,7,10,13,16,2,5,8,11,14,17,3,6
    ,9,12,15,18                                  <210>
750 DATA 32,253,174,32,158,183,32,30,225,3
    2,253,174,32,158,183,138,72,32,253           <246>
760 DATA 174,32,139,176,133,73,132,74,32,1
    63,182,104,32,117,180,160,2,185              <230>
770 DATA 97,0,145,73,136,16,248,200,32,18,
    225,145,98,200,196,97,208,246,32             <080>
780 DATA 204,255,96,0,0,0,0,0,32,245,207,1
    38,72,32,245,207,224,0,176,3,76              <208>
790 DATA 72,178,224,24,176,249,134,251,32,
    245,207,224,0,144,240,224,39,176             <038>
800 DATA 236,134,253,32,245,207,224,25,176
    ,227,134,252,232,138,56,229,251              <138>
810 DATA 144,218,240,216,133,250,32,245,20
    7,224,40,176,207,228,253,144,203             <144>
820 DATA 134,254,104,170,165,172,72,165,17
    3,72,165,174,72,165,175,72,224,0             <064>
830 DATA 208,22,166,251,198,250,240,44,32,
    240,233,232,189,240,236,133,172              <188>
840 DATA 181,217,32,219,207,48,236,202,240
    ,3,76,72,178,166,252,198,250,240             <028>
850 DATA 16,32,240,233,202,189,240,236,133
    ,172,181,217,32,219,207,48,236,164           <090>
860 DATA 254,32,240,233,32,36,234,169,32,1
    45,209,136,196,253,16,249,76,88              <224>
870 DATA 233,41,3,13,136,2,133,173,32,224,
    233,164,254,177,172,145,209,177              <206>
880 DATA 174,145,243,136,196,253,16,243,96
    ,32,253,174,76,158,183                       <209>
```

Listing zum »Directory-Sorter«. Beachten Sie bitte den Checksummer auf Seite 6



POKEs, die Sie kennen sollten

Welcher C 64-Benutzer hat sich noch nicht mit dem Basic seines Computers herumgeärgert? Viele Programmierlösungen sind nur über PEEK und POKE möglich. Die folgende Liste der wichtigsten PEEKs und POKEs soll Ihnen behilflich sein.

Hex	Dezimal	Beschreibung
1	1	Inhalt 55 = normal Inhalt 54 = Basic ausgeschaltet (auf RAM umgestellt) Inhalt 53 = Basic und Kernal auf RAM umgestellt. (Es empfiehlt sich dabei, das Basic und das Kernel vorher ins RAM zu POKEn, damit der Computer bei der Umschaltung nicht aussteigt.)
A	10	Ist PEEK (10) = 0, so war der letzte Befehl LOAD. Ergibt PEEK (10) eine 1, ist VERIFY eingegeben worden.
11	17	Mit diesem PEEK läßt sich abfragen, wie die letzte Variable zugewiesen wurde. Ist PEEK (17) = 00, dann war die letzte Variablenzuweisung ein INPUT, oder es hat noch keine Zuweisung stattgefunden. Ist PEEK (17) = 64, dann wurde die letzte Variable durch GET geholt. Bei PEEK (17) = 152 erfolgte die letzte Variablenübergabe durch einen READ-Befehl.
13	19	Durch POKE 19,64 wird beim nächsten INPUT-Befehl kein Fragezeichen mehr ausgegeben. Allerdings kann man nachher durch Drücken der RETURN-Taste nicht mehr in die nächste Zeile gelangen. Es empfiehlt sich daher, nach dem INPUT-Befehl diesen Befehl wieder mit POKE 19,0 rückgängig zu machen.
18	24	Mit POKE 24,0 wird FORMULAR TOO COMPLEX ERROR aufgehoben.
2B/2C	43/44	Der Anfang des zur Zeit im Speicher befindlichen Basic-Programms errechnet sich durch PEEK (43)+PEEK (44) *256.
2D/2E	45/46	Das Ende des Basic-Programms erhält man durch PEEK (45)+PEEK (46)* 256.
37/38	55/56	Mit PEEK (55)+256*PEEK (56) kann das Ende des Basic-RAMs abgerufen werden.
39/3A	57/58	Die Zeilennummer, bei der nach einer Programmunterbrechung gestoppt wurde, errechnet sich durch PEEK (57)+256*PEEK (58).
3D/3E	61/62	Zeiger auf Basic-Statement für CONT: Durch PEEK (61)+PEEK (62)*256 erhält man die Speicherstelle, die nach dem zuletzt ausgeführten Basic-Befehl liegt, das heißt die Speicherstelle, von der sich der Basic-Interpreter bei CONT den nächsten Befehl holt. Tip: Bei CONT kommt öfter CAN'T CONTINUE ERROR vor, wenn man nach dem Stoppen ein CLR eingegeben oder in irgendeiner Programmzeile etwas geändert hat. Liest man die Werte mit PEEK (61) und PEEK (62) nach der Unterbrechung aus, dann macht ein CLR oder ähnliches nichts aus, wenn man vor CONT die zuvor ausgelesenen Werte wieder in die Speicherstellen POKEt.
3F/40	63/64	Nummer der aktuellen DATA-Zeile: Mit PEEK (63)+PEEK (64)*256 erhält man die Nummer der DATA-Zeile, aus der gerade das letzte Datum geholt wurde. (Gut zum Finden von Fehlern in DATA-Zeilen geeignet.)
45/46	69/70	Zuletzt zugewiesene Variable: Bei normalen Fließkommavariablen liest man den Wert mit PRINT CHR$(PEEK (69))+CHR$(PEEK (70)) aus. Bei Integervariablen (zum Beispiel XY%) erhält man den Namen durch PRINT CHR$(PEEK (69)-128)+CHR$ (PEEK (70)-128). Strings (zum Beispiel VX$) erhält man durch PRINT CHR$(PEEK(69))+ CHR$(PEEK (70)-128).
78	120	Nach Ausführung dieses POKEs nimmt der C 64 keinerlei Befehle mehr an: POKE 120,2
90	144	Statusbyte: Mit PRINT ST läßt sich das Statusbyte abfragen.
91	145	Nach WAIT 145,16,16 wartet der Computer auf den Feuerknopf von Joystick 1.
93	147	Wenn man die LOAD-Routine im Betriebssystem anspringt, holt es sich aus der Speicherstelle 147 die Information, ob LOAD oder VERIFY durchgeführt wird. Inhalt 0 = LOAD Inhalt 4 = VERIFY
9D	157	Ausgabe-Kontrolle: Inhalt 000 = Progamm-Modus Inhalt 128 = Direktmodus Damit bei LOAD-Befehlen vom Programm aus die Mitteilungen SEARCHING, LOADING oder VERIFYING auf dem Bildschirm erscheinen, setzt man vor dem LOAD- VERIFY- oder SAVE-Befehl ein POKE157,128.
B6	182	PEEK (182) ruft die Anzahl der Zeichenlesefehler ab.
B7	183	PEEK (183) gibt die Länge des Filenamens an.
B8	184	Die laufende Filenummer kann mit PEEK (184) abgerufen werden.
B9	185	Die aktuelle Sekundäradresse kann mit PEEK (185) angegeben werden.
BA	186	Die derzeitige Gerätenummer wird

Hex	Dezimal	Beschreibung
		durch PEEK (186) abgerufen.
C5	197	Derzeitiger Tastendruck: PEEK (197)
C6	198	Nach WAIT 198,1 wartet der Computer auf eine gedrückte Taste.
C7	199	RVS Flag: eine reverse Zeichenausgabe bei PRINT erfolgt nach POKE 199,1. POKE 199,0 hebt den RVS-Modus auf.
C8	200	Zeiger auf Zeilenende. PEEK (200) gibt an, wieviel Zeichen die zuletzt eingegebene Zeile hatte.
C9/CA	201/202	PEEK (201) gibt die Zeile der aktuellen Cursorposition an, während PEEK (202) die Spalte abruft.
CB	203	Die derzeitig gedrückte Taste im CHR$-Modus kann mit PEEK (203) abgefragt werden. Ist keine Taste gedrückt, so steht der Wert 64 in diesem Byte.
CC	204	Nach POKE 204,0 bleibt der Cursor an, auch bei GET-Befehlen. Mit POKE 207,0 : POKE 204,1 kommt man dann wieder auf den Normalzustand zurück.
D3	211	POKE 211,Wert setzt die Spalte der Cursorposition. Werte von 0 bis 39 können eingegeben werden. Siehe auch 214.
D5	213	Länge der momentanen Bildschirmzeile: über PEEK (212) läßt sie sich abfragen.
D6	214	POKE 214,Wert setzt die Zeile der Cursorposition. Die Werte 0–24 können eingegeben werden. Nach SYS 58640 erscheint der Cursor auf der entsprechenden Position.
D7	215	Das zuletzt eingegebene Zeichen im ASCII-Code liest man mit PEEK (215) aus.
277–280	631–640	Tastaturpuffer (ASCII-Code): Über den POKE-Befehl lassen sich beispielsweise Zeichen in den Tastaturpuffer eingeben.
281–284	641–644	Start- und Endadresse des Basic-RAMs: Durch Ändern dieser Werte kann man die Größe des Basic-RAMs verändern, zum Beispiel: POKE 643,0 : POKE 644,128 : SYS 64764 setzt das Ende des Basic-RAMs um 8 KByte nach unten. Anderes Beispiel: POKE 641,0 : POKE 642,16 : SYS 64764 setzt das Basic-RAM um 2 KByte nach oben.
286	646	POKE 646,Farbwert setzt die Cursorfarbe.
287	647	Farbe unter dem Cursor: Mit den Werten 0 bis 15 werden die jeweiligen Farben gePOKEt.
289	649	Länge des Tastaturpuffers: Nach POKE 649,0 wird die gedrückte Taste nicht mehr auf den Bildschirm ausgegeben. Die maximale Größe des Tastaturpuffers beträgt 10.
28C	652	Zähler für REPEAT-Verzögerung: Alle eingePOKEten Werte verzögern die Repeat-Funktion. Höherer Wert = größere Verzögerung.
28D	653	PEEK für Shift-, Commodore- und für CTRL-Taste: Bit 0 = Shift-Taste, Bit 1 = Commodore-Taste und Bit 2 = Control-Taste.
291	657	Nach POKE 657,128 ist eine Umschaltung Groß-/Kleinschrift von der Tastatur her unmöglich. POKE 657,0 erzeugt wieder den Normalzustand.
302	770	Durch EinPOKEn eines beliebigen Wertes erfolgt die READY-Ausgabe unendlich oft, nur noch Ausschalten hilft.
307	775	POKE 775,Wert aktiviert einen Listschutz. Nur über den Wert 167 kann der Listschutz aufgehoben werden.
308	776	Der Befehl POKE 776,1 zerstört das Programm.
309	777	Nach POKE 777,1 wird kein Befehl mehr ausgeführt. Der Cursor befindet sich in der linken Ecke.
30D/30E	781/782	Startadresse, ab der ein Programm geladen wird: Durch entsprechende POKE-Werte kann ein Basic-Programm in einen anderen Speicherbereich geladen werden.
310/311	784/785	USR-Vektor: Erfolgt der Einsprung in ein Maschinenprogramm über den USR-Befehl, so muß die Einsprungadresse vorher in diese beiden Byte gePOKEt werden.
314/315	788/789	IRQ, Hardware-Interrupt: Das Betriebssystem springt ständig in diese Routine, durch Ändern des Inhalts kann man eigene, »interruptgesteuerte« Maschinenroutinen ständig laufen lassen. POKE 788,49 hebt die Wirkung der Stop-Taste auf. POKE 788,52 schaltet sie wieder ein.
318/319	792/793	Restore-Vektor: PEEK (792)+PEEK (793)*256 ergibt die Speicherstelle, an die bei Restore-Tastendruck gesprungen wird. Beispiel: Bei POKE 792,226 : POKE 793,252 wird beim Drücken der Restore-Taste ein Reset ausgelöst.
321/322	801/802	Durch die Kombination POKE 801,0 : POKE 802,0 : POKE 818,165 wird ein SAVE-Schutz aktiviert.
328	808	POKE 808,225 hebt die Stop-Taste auf, POKE 808,237 reaktiviert sie.
32D	813	Nach POKE 813,2 läßt sich ein Programm nicht mehr ändern.
332/333	818/819	Folgende Kombinationen bewirken ebenfalls einen SAVE-Schutz: 1. POKE 818,116 : POKE 819,196 2. POKE 818,34 : POKE 819,253
33C–3FB	828–1019	Kassettenpuffer: Nach LOAD oder VERIFY stehen im Kassettenpuffer folgende Informationen: 828: 1 = normales File, 3 = wurde mit SAVE"Name",1,1 abgespeichert. Solche Programme werden bei LOAD automatisch ab der Adresse geladen, von der sie abgespeichert wurden. 829/830: Hier ist die Startadresse des Programms abgelegt (829 ist das Low-Byte, 830 das High-Byte). 831/832: Endadresse des Programms. 833–1019: 186 Zeichen langer Programmname (auf dem Bildschirm werden nur 16 angezeigt, aber es lassen

64er-online.de

Hex	Dezimal	Beschreibung
		sich bis 186 Stellen lange Programmnamen abspeichern). Der Kassettenpuffer ist auch gut zum Ablegen eigener Maschinenprogramme geeignet, sofern mit der Floppy gearbeitet wird.
9006	36870	PEEK (36870) liest die Horizontal-Position des Lichtgriffels.
9007	36871	PEEK (36871) liest die Vertikal-Position des Lichtgriffels.
	42291	Koppeladressen angleichen: Falls Programme mit NEW gelöscht wurden, kann man mit diesem SYS-Befehl die Byte 2049 und 2050 wieder in Ordnung bringen, wenn vorher etwas anderes als 0 in diese Speicherzellen gePOKEt wird.
DO16	53270	VIC-Control-Register: Ein horizontales Softscrolling kann gePOKEt werden. POKE 53270,0 = scrollen nach links POKE 53270,1 = scrollen nach rechts
DO20	53280	POKE 53280,Wert setzt die Rahmenfarbe des Bildschirms.
DO21	53281	POKE 53281,Wert setzt die Hintergrundfarbe auf dem Bildschirm.
DC00	56320	Joystick Port 2: WAIT 56320,16,16 wartet auf Feuerknopf WAIT 56320,4,4 wartet auf Linksbewegung des Joysticks WAIT 56320,1,1 wartet auf Joystick nach oben WAIT 56320,2,2 wartet auf Joystick nach unten WAIT 56320,8,8 wartet auf Joystick nach rechts
DC01	56321	Wie 56320, aber Joystick in Port 1.
DC05	56325	Die Blinksequenz des Cursors kann in diesem Byte bestimmt werden. Höherer Wert = langsamere Blinksequenz. Mit POKE 56325,58 wird die normale Blinksequenz festgelegt.
DC90	56464	Nach WAIT 56464,16,16 wartet der Computer auf den Feuerknopf des Joysticks 2.
DD00	56576	Mit PEEK(56576) kann man die Pins PB0–PB7 vom User-Port (auf der Unterseite des Ports, siehe Handbuch) auslesen. Mit POKE in diese Speicherstelle kann man auch Ausgaben über den User-Port laufen lassen.
DD02	56578	Datenrichtungsregister für User-Port: Jedes der Bits gibt die Datenrichtung für die Pins PB0–PB7 des User-Ports an (Bit gesetzt = Ausgang, Bit nicht gesetzt = Eingang).
FF81	65409	SYS 65409 setzt den Video-Chip des C64 auf den Ursprungszustand zurück.
FFD5	65493	LOAD-Routine des Betriebssystems. Mit folgender kleiner Routine kann man Unterprogramme nachladen, ohne irgendwelche Basic-Pointer (wie zum Beispiel die Zeiger auf die Endadresse, 45 und 46) zu verändern: POKE 186,1 : POKE 780,0 : POKE 781,0 : POKE 782,96 : POKE 183,0 : SYS 65493 Erklärung: 186,1 = Geräteadresse für Recorder 781 und 782 gibt die Startadresse an, ab der das Programm geladen werden soll. 183,0 = kein Programmname. SYS 65493 = LOAD-Routine.
FFE7	65511	Durch SYS 65511 lassen sich alle Files schließen. So erspart man sich das lästige Eintippen von CLOSE1:CLOSE2:CLOSE3... Dabei sollte aber beachtet werden, daß auf diese Weise nur der Kanal geschlossen wird, aber keine Dateien auf einer Disk.

Grafik und Sprites wurden im Grafikkurs von H. Ponnath und im 64'er Sonderheft 4 sehr ausführlich beschrieben. Auf PEEKs und POKEs für diesen Bereich wurde daher verzichtet.

(M. Kohlen/kn)

Die Modulfabrik

Ganz gleich, ob Sie ein Maschinensprache- oder Basic-Programm in einem EPROM speichern möchten – mit dem Modulgenerator geht es blitzschnell und komfortabel.

Der Modulgenerator zeichnet sich gegenüber vielen zum Teil mit der Hardware (EPROM-Programmiergerät) mitgelieferten und anderen käuflichen Programmen dieser Art durch seine einfache, menügesteuerte Bedienung aus. Er ermöglicht das komfortable Generieren eines aus bis zu fünf Programmen (Basic oder Maschinensprache) bestehenden Auto-Start-Moduls mit Menüsteuerung. Dieses Modul kann anschließend mit der Treibersoftware des EPROM-Brenners auf ein EPROM übertragen werden.

Vor der Beschreibung des Modulgenerators selbst sollen zuvor der Aufbau und die Funktionsweise der von ihm generierten Programme erläutert werden. Sie belegen den Bereich von $8000 bis $A000 und eignen sich somit zum Brennen auf ein 0, beziehungsweise $AFFF, 8 KByte oder 16 KByte EPROM. Hinter der Modulkennung »CBM 80« für den Autostart befindet sich ein kurzes Steuerprogramm zur Tastaturabfrage, zum Verschieben und Starten der gespeicherten Programme; dahinter sind eine Liste der Programmnamen sowie eine Tabelle mit wichtigen Adressen:

Byte 1 und 2:	Anfangsadresse des Programms im Modul
Byte 3 und 4:	Endadresse+1 des Programms im Modul
Byte 5 und 6:	Endadresse+1 des Programms nach dem Verschieben
Byte 7 und 8:	Startadresse des Programms

Zum Verschieben der Programme aus dem EPROM in den eigentlichen Speicherbereich wird die Blockverschiebe-Routine des Basic-Interpreters ($A3BF) benutzt. Hierbei sind die ersten sechs Byte erforderlich.

Zum Start des Programms werden zunächst die Bytes 7 und 8 auf Basic-Start geprüft ($0801). Bei positivem Ergebnis wird später ein »RUN«-Befehl simuliert.

Ist es negativ, wird geprüft, ob diese Bytes gleich Null sind. In diesem Fall wird das Programm nicht gestartet, und man bleibt weiterhin im Modul-Menü. Auf diese Weise können mehrteilige Programme verschoben werden, bevor das eigentliche Hauptprogramm gestartet wird (enthaltene Nachlade-Befehle müssen bei diesem Programmteil selbstverständlich vorher entfernt werden). Enthalten die Bytes 7 und 8 weder die Basic-Kennung noch die Null-Kennung, wird das Programm direkt angesprungen, das heißt mit jmp $xxxx gestartet.

Hinter den beschriebenen Tabellen befinden sich die gespeicherten Programme unmittelbar hintereinander. Tabelle 1 zeigt noch einmal schematisch den beschriebenen Aufbau:

$8000	Modulkopf »CBM 80« etc.
$8009	Modulname
$8033	Initialisierung nach Reset
$805C	Steuerprogramm mit Verschiebe- und Startroutine
$8149	Text des Modul-Menüs (ASCII-Zeichen)
$8173	Programmnamen (maximal fünf je 16 Zeichen)
$81C3	Tabelle (Adressen)
$81EB	Programm 1
	.
	.
	.
	Programm n
$9FFF	letztes Byte
$A000	----------------------

Tabelle 1. Aufbau der generierten Modul-Programme

Durch das notwendige Steuerprogramm stehen dem Anwender bei 8-KByte-Modulen lediglich 30 oder 32 Blöcke für die Programme zur Verfügung, was für die meisten Anwendungen ausreichend ist.

Nachdem Aufbau und Funktionsweise der generierten Programme kurz erläutert wurden, folgt nun die Beschreibung des Modulgenerators. Er belegt das von Basic nicht benutzte RAM von $C000–$CFFF. Nach dem Kopieren des eben beschriebenen Modul-Steuerprogramms nach $8000 befindet man sich in einem Menü, das folgende Programmpunkte enthält:

F1 Directory
F3 Disc Commands
F5 File Load
F7 Save Modul
F8 Quit

Nach der Betätigung einer der Funktionstasten prüft der Modulgenerator, ob die Diskette (Gerätenummer 8) eingeschaltet ist. Wenn nicht, ertönt ein akustisches Signal, und es wird eine Fehlermeldung ausgegeben (»Device not present«). Bei erfolgreichem Test wird das gewünschte Menü-Unterprogramm gestartet. Jedes Unterprogramm ist durch Eingabe eines »RETURN« zu verlassen, so daß man sich die einzelnen Programmpunkte ruhig ansehen kann.

Der erste Programmpunkt bedarf sicherlich keiner näheren Erläuterung.

Das zweite, mit der Funktionstaste F3 gestartete Unterprogramm, dient dem Senden von Floppy-Systembefehlen (Scratch, Rename, Copy, Format, Validate, Initial., etc.), mit dem auch das Listen von Teilen der Directory möglich ist, zum Beispiel:

»$:T?A*« listet alle Programme des Directory, deren erster Buchstabe ein »T« und deren dritter Buchstabe ein »A« ist.

Die Eingabe des Floppy-Kommandos muß im Modulgenerator ohne Anführungsstriche erfolgen.

Das dritte Unterprogramm des Modulgenerators (F5) dient dem Laden beziehungsweise Anhängen (Merge) eines Programms an das Modul-Programm (ab Adresse $81EB), beziehungsweise an das zuletzt geladene. Wird dieses Unterprogramm gewählt, werden auf dem Bildschirm die Anzahl freier Bytes, die Anzahl freier Blöcke und die Anzahl der bereits geladenen Programme angezeigt. Ist diese Zahl bereits fünf, erscheint kurz die Meldung »Out of Memory«, und man befindet sich wieder im Hauptmenü. Andernfalls erscheint »Filename:«, und man kann den Namen des gewünschten Programms eingeben. Drückt man dagegen nur die »RETURN«-Taste, gelangt man, wie bereits erwähnt, ins Hauptmenü zurück. Nach der Eingabe des Filenamens und »RETURN« wird versucht, das Programm zu laden. Tritt dabei ein Fehler auf (zum Beispiel File not found), wird dies unverzüglich angezeigt und zum Hauptmenü zurückgesprungen. Benötigt das geladene Programm mehr Speicherkapazität als im Modul vorhanden, wird dies hinterher mit »Out of Memory« angezeigt und das Programm als nicht geladen betrachtet.

Nach erfolgreichem Ladevorgang (Meldung »00,OK,00, 00«) fragt das Programm, ob die Ladeanfangsadresse auch die Startadresse ist. Bei Ladeanfangsadresse $0801 fragt er, ob das Programm mit »RUN«, andernfalls ob es mit »SYS Anfangsadresse« gestartet werden soll. Diese Frage ist mit »J« oder »N« zu beantworten. Wird die Frage bestätigt, erscheint »O.K.«, und man befindet sich wieder im Hauptmenü. Bei »N« wird man nach der dezimalen (Integer-Zahl) Startadresse des Files gefragt (»Start mit SYS:?«). Wird an dieser Stelle lediglich die »RETURN«-Taste gedrückt, wird das Programm später im Modul lediglich verschoben (nicht gestartet!). Die Filenamen und die Adressen (Anfang, Ende und Start) werden in die Tabellen des Modul-Programms eingetragen, bevor in das Hauptmenü des Modulgenerators zurückgesprungen wird. Ist das Modul-Programm erstellt, kann es mit dem vierten Menüpunkt (F7) abgespeichert werden. Zunächst wird wiederum nach dem Filenamen gefragt, unter dem das Programm auf der Diskette abgelegt werden soll. Der eingegebene Filename des Moduls erscheint später auch mit dem Zusatz »Modul-Version« in der obersten Bildschirmzeile des Modul-Menüs und sollte deshalb sinnvoll gewählt werden.

Gespeichert wird immer der gesamte Bereich von $8000 bis $A000 (beziehungsweise bis $AFFF), so daß 33 Blöcke (66 Blöcke) auf der Diskette belegt werden.

Nach dem Speichern kann man mit dem letzten Punkt des Menüs (F8) den Modulgenerator verlassen. Wurde ein Programm geladen, das Modul jedoch nicht gespeichert, erscheint der Hinweis »Warning – Changes Not Saved. Quit (J/N)«. Bei Eingabe von »J« wird man gefragt, ob man das Modul testen oder zurück zu Basic will. Beim Modultest wird lediglich ein Software-Reset ausgelöst, mit dem man direkt in das erstellte Modul-Programm gelangt und die integrierten Files mit den Tasten 1–n (n=maximal 5) starten kann.

Bei »Quit TO Basic« wird das Programm mit einem »rts« (entspricht dem Basic-Befehl »end« verlassen. Der Modulgenerator befindet sich weiterhin im RAM (Basic-RAM und ab Adresse $C000) und kann deshalb sowohl mit »RUN« als auch mit »SYS 49152« neu gestartet werden.

Einen Warmstart des Modulgenerators (Start ohne Verlust der bereits geladenen Files) erreicht man durch Eingabe von: »SYS 49175«.

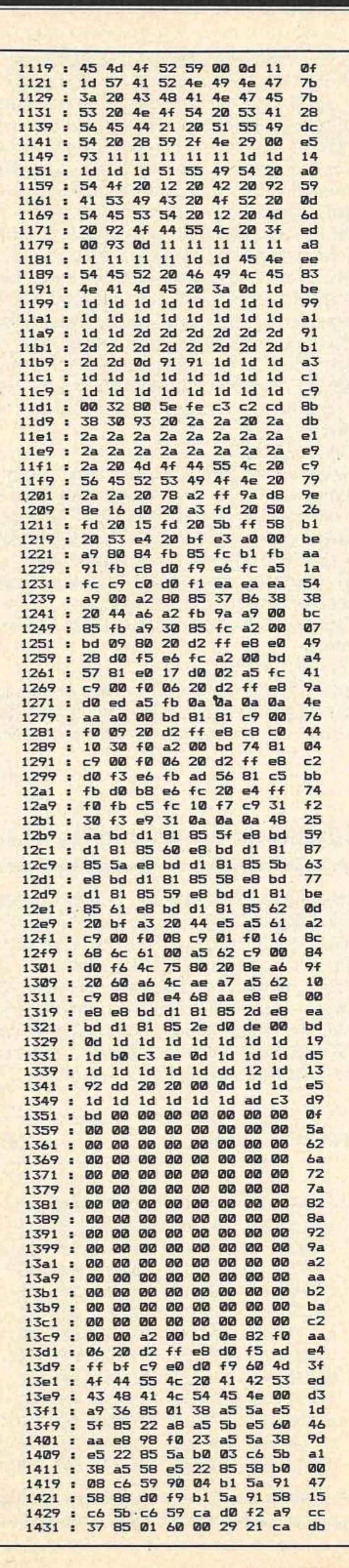
Der Modulgenerator (Listing) wird mit dem MSE eingegeben. (Norbert Jungmann/aw)

```
programm : modulgen           0801 1436
---------------------------------------

0801 : 0d 08 c1 07 9e 28 32 30  b8
0809 : 36 33 29 00 00 00 a9 2d  24
0811 : 85 5f a9 08 85 60 a9 36  20
0819 : 85 5a a9 14 85 5b a9 59  45
0821 : 85 58 a9 cf 85 59 20 bf  5a
0829 : a3 4c 50 c3 a9 00 8d fd  4c
0831 : cf a9 ff 8d fc cf a9 05  85
0839 : 8d 20 d0 a9 00 8d 21 d0  d2
0841 : 20 62 c8 a2 00 86 fb bd  b8
0849 : ab ca e8 e0 00 d0 03 ee  20
0851 : 6d c3 c9 00 f0 06 20 d2  78
0859 : ff 4c 6b c3 e6 fb a5 fb  ae
0861 : c9 10 f0 0a 8a 48 20 3d  96
0869 : c4 68 aa 4c 6b c3 20 3e  67
0871 : f1 f0 fb c9 85 f0 1f c9  02
0879 : 87 f0 12 c9 88 f0 14 c9  2a
0881 : 8c f0 0d c9 86 f0 03 4c  96
0889 : 92 c3 4c e2 c7 4c 44 c4  e6
0891 : 4c a6 c6 4c 05 c7 20 b4  e4
0899 : c7 a9 93 20 d2 ff a9 24  3a
08a1 : 85 fb a9 fb 85 bb a9 00  ea
08a9 : 85 bc a9 01 85 b7 a9 08  e4
08b1 : 85 ba a9 60 85 b9 20 d5  5c
08b9 : f3 a9 08 20 b4 ff a9 60  3a
08c1 : 20 96 ff a9 00 85 90 a0  11
08c9 : 03 84 fb 20 a5 ff 85 fc  7c
08d1 : a4 90 d0 2f 20 a5 ff a4  50
08d9 : 90 d0 28 a4 fb 88 d0 e9  8b
08e1 : a6 fc 20 cd bd a9 20 20  b1
08e9 : d2 ff 20 a5 ff a6 90 d0  91
08f1 : 12 aa f0 06 20 d2 ff 4c  86
08f9 : 0e c4 a9 0d 20 d2 ff a0  4f
0901 : 02 d0 c6 20 42 f6 a9 b1  07
0909 : a0 c9 20 1e ab 20 3e f1  f2
0911 : f0 fb c9 0d d0 f7 4c 67  e0
0919 : c3 a2 0b 86 d3 4c 6c e5  de
0921 : 20 b4 c7 ad fd cf 69 2f  a5
0929 : 8d 55 ca a9 d0 a0 c9 20  c2
0931 : 1e ab a9 00 38 ed fe cf  1e
0939 : aa a9 a0 ed ff cf 85 fb  2a
0941 : 86 fc 20 cd bd a2 02 a0  41
0949 : 22 18 20 0a e5 a6 fc a5  93
0951 : fb 85 62 86 63 a2 90 38  76
0959 : 20 49 bc 20 0c bc a9 01  a0
0961 : a2 00 85 62 86 63 a2 90  e0
0969 : 38 20 49 bc 20 12 bb 20  5d
0971 : e2 ba 20 49 b8 20 cc bc  1b
0979 : 20 fe ba 20 df bd 20 1e  74
0981 : ab ad fd cf c9 05 d0 0b  9a
0989 : a2 11 a0 00 18 20 0a e5  52
0991 : 4c cc c5 a9 9e a0 cc 20  4c
0999 : 1e ab a2 00 20 cf ff c9  4a
09a1 : 0d f0 07 9d e0 cf e8 4c  65
09a9 : c0 c4 8a 48 a9 00 a2 08  ac
09b1 : a0 00 20 ba ff 68 a2 e0  40
09b9 : a0 cf 20 bd ff ae fe cf  12
09c1 : ac ff cf 8e d0 cf 8c d1  94
09c9 : cf a9 00 85 93 85 90 a4  0f
09d1 : b7 d0 03 4c 67 c3 a9 60  37
09d9 : a6 b9 85 b9 20 d5 f3 a5  c0
09e1 : ba 20 b4 ff a5 b9 20 96  ae
09e9 : ff 20 a5 ff 8d d6 cf 8d  4b
09f1 : d4 cf a5 90 4a 4a b0 4d  7d
09f9 : 20 a5 ff 8d d7 cf 8d d5  7b
0a01 : cf ad fe cf 85 ae ad ff  e5
0a09 : cf 85 af a9 fd 25 90 85  12
0a11 : 90 20 e1 ff d0 06 20 33  4e
0a19 : f6 4c 67 c3 20 a5 ff aa  0c
0a21 : a5 90 4a 4a b0 e5 8a a0  90
0a29 : 00 91 ae e6 ae d0 02 e6  c1
0a31 : af ee d4 cf d0 03 ee d5  13
0a39 : cf 24 90 50 ce 20 ab ff  e5
0a41 : 20 42 f6 90 09 20 81 c7  79
0a49 : 20 83 c5 4c 6c c6 18 a6  d0
0a51 : ae a4 af 8e d2 cf 8c d3  94
0a59 : cf 20 83 c5 4c ad c5 a2  60
0a61 : 00 bd ca cb e0 0a f0 06  9a
0a69 : 20 d2 ff e8 d0 f3 a9 08  73
0a71 : 85 ba 20 b4 ff a9 6f 85  08
0a79 : b9 20 96 ff 20 a5 ff 20  57
0a81 : d2 ff c9 0d d0 f6 4c ab  b4
0a89 : ff ad d3 cf c9 a0 10 18  60
0a91 : ad d6 cf c9 01 d0 3b ad  b5
0a99 : d7 cf c9 08 d0 34 a9 aa  76
0aa1 : a0 cb 20 1e ab 4c dc c5  0f
0aa9 : 20 81 c7 20 3d c4 a9 2e  7d
0ab1 : a0 cc 20 1e ab 4c 6c c6  e0
0ab9 : 20 e4 ff f0 fb c9 59 f0  be
0ac1 : 0e c9 4e d0 f3 4c 82 c6  9b
0ac9 : a9 ca a0 cb 4c 1e ab 4c  76
0ad1 : 19 c6 a9 e4 a0 cb 20 1e  79
0ad9 : ab ad d7 cf ae d6 cf 20  6c
0ae1 : cd bd 20 e4 ff f0 fb c9  3c
0ae9 : 59 f0 0a c9 4e f0 03 4c  87
0af1 : 06 c6 4c 82 c6 20 24 c6  49
0af9 : a9 00 8d fc cf 4c 6c c6  44
0b01 : ad fd cf 0a 0a 0a aa a0  bf
0b09 : 00 b9 d0 cf 9d d1 81 e8  54
0b11 : c8 c0 08 d0 f4 ae fd cf  b2
0b19 : 8a 0a 0a 0a 0a aa a0 00  e5
0b21 : b9 e0 cf 9d 81 81 c8 e8  0b
0b29 : c4 b7 30 f4 ee fd cf ad  ed
0b31 : fd cf 8d 56 81 ad d2 cf  b4
0b39 : 8d fe cf ad d3 cf 8d ff  e1
0b41 : cf a9 19 a0 cc 4c 1e ab  3e
0b49 : 18 a9 05 a0 ff a2 ff ca  36
0b51 : d0 fd 88 d0 fa e9 01 c9  f2
0b59 : 00 d0 f4 4c 67 c3 a9 ca  59
0b61 : a0 cb 20 1e ab 20 60 a5  3b
0b69 : 84 7b 86 7a 20 73 00 20  7a
0b71 : f3 bc 20 f7 b7 a5 14 a6  10
0b79 : 15 8d d6 cf 8e d7 cf 4c  84
0b81 : 19 c6 ad fc cf c9 00 f0  35
0b89 : 07 ad fd cf d0 1f f0 41  2c
0b91 : 20 81 c7 a9 42 a0 cc 20  35
0b99 : 1e ab 20 e4 ff f0 fb c9  3c
0ba1 : 59 f0 0a c9 4e f0 03 4c  3f
0ba9 : be c6 4c 67 c3 ad 58 c7  65
0bb1 : c9 c0 f0 1d a9 6c a0 cc  d4
0bb9 : 20 1e ab 20 e4 ff f0 fb  e1
0bc1 : c9 42 f0 0d c9 4d f0 11  76
0bc9 : c9 0d f0 02 d0 ed 4c 67  12
0bd1 : c3 a9 00 8d 05 80 4c 31  03
0bd9 : a8 6c fc ff a9 37 85 01  63
0be1 : 60 20 b4 c7 a9 9d a0 cc  1b
0be9 : 20 1e ab a2 00 20 cf ff  98
0bf1 : c9 0d f0 12 8d 0e 80 e8  dc
0bf9 : 20 cf ff c9 0d f0 0a 9d  f6
0c01 : 0e 80 e8 4c 1c c7 4c 67  13
0c09 : c3 8a 48 a9 20 9d 0e 80  81
0c11 : a9 01 a2 08 a0 00 20 ba  e4
0c19 : ff 68 a2 0e a0 80 20 bd  c1
0c21 : ff a9 36 85 01 a9 00 85  9c
0c29 : 61 85 9d a9 80 85 62 a9  fa
0c31 : 61 a2 00 a0 a0 20 d8 ff  66
0c39 : 90 0d 20 00 c7 20 81 c7  6b
0c41 : 20 83 c5 18 4c 67 c3 20  e7
0c49 : 00 c7 a9 ff 8d fc cf 4c  30
0c51 : 67 c3 a9 f5 85 5f a9 cc  56
0c59 : 85 60 4c bf a3 a9 09 8d  e0
0c61 : 05 d4 a9 80 8d 06 d4 a9  fa
0c69 : 0f 8d 18 d4 a9 10 8d 01  33
0c71 : d4 8d 00 d4 a9 11 8d 04  08
0c79 : d4 a2 00 a0 96 [illegible] 10 fd  b1
0c81 : 88 d0 fa 8c 04 d4 8c 05  e5
0c89 : d4 8c 00 d4 8c 01 d4 60  23
0c91 : a9 0f a2 08 a0 0f 20 ba  e4
0c99 : ff a9 00 20 bd ff 20 c0  4f
0ca1 : ff a9 0f 20 c3 ff 20 b7  69
0ca9 : ff d0 01 60 20 81 c7 68  5b
0cb1 : 68 20 3d c4 a9 92 a0 c9  56
0cb9 : 20 1e ab 4c 92 c3 20 b4  8e
0cc1 : c7 a9 93 20 d2 ff a9 6e  f6
0cc9 : a0 c9 20 1e ab a2 00 20  2a
0cd1 : cf ff c9 0d f0 06 c9 24  62
0cd9 : f0 05 d0 06 4c 67 c3 4c  e8
0ce1 : 43 c8 20 0c c8 4c 1f c8  0f
0ce9 : 9d e0 cf e8 20 cf ff c9  1b
0cf1 : 0d f0 07 9d e0 cf e8 4c  b5
0cf9 : 10 c8 60 8a 48 a9 0f a2  2a
0d01 : 08 a0 0f 20 ba ff 68 a2  b4
0d09 : e0 a0 cf 20 bd ff 20 c0  0f
0d11 : ff a9 0f 20 c3 ff 20 81  6c
0d19 : c7 20 83 c5 4c 6c c6 20  0d
0d21 : 0c c8 8a 85 b7 a9 93 20  3c
0d29 : d2 ff a9 e0 85 bb a9 cf  fe
0d31 : 85 bc a9 01 a2 08 a0 60  4d
0d39 : 20 ba ff 4c da c3 a9 ab  09
0d41 : a0 ca 20 1e ab 20 3d c4  4c
0d49 : a2 00 a0 00 bd 14 c9 f0  99
0d51 : 06 20 d2 ff e8 d0 f5 e8  db
0d59 : c8 c0 07 f0 0e 8a 48 98  e9
0d61 : 48 20 3d c4 68 a8 68 aa  64
0d69 : 4c 70 c8 20 e4 ff f0 fb  2d
0d71 : c9 31 f0 43 c9 32 f0 02  6d
0d79 : d0 f1 a9 c0 8d b1 c5 8d  5d
0d81 : 58 c7 8d 5e c4 a9 20 a2  4b
0d89 : f9 a0 81 8d 59 cd 8e 5a  d7
0d91 : cd 8c 5b cd a9 1f a2 82  58
0d99 : 8d 0d ce 8e 0e ce a9 63  f7
0da1 : a2 82 8d fe cf 8e ff cf  d8
0da9 : 85 58 86 59 a9 58 a2 cf  af
0db1 : 85 5a 86 5b 4c 76 c7 a9  5b
0db9 : a0 8d b1 c5 8d 58 c7 8d  1b
0dc1 : 5e c4 a9 ea 8d 59 cd 8d  3f
0dc9 : 5a cd 8d 5b cd a9 bf a2  47
0dd1 : a3 8d 0d ce 8e 0e ce a9  40
0dd9 : f9 a2 81 8d fe cf 8e ff  de
0de1 : cf 85 58 86 59 a9 ee a2  3e
0de9 : ce 85 5a 86 5b 4c 76 c7  63
0df1 : 45 50 52 4f 4d 54 59 50  5a
0df9 : 3a 0d 0d 00 12 b0 c3 ae  10
0e01 : 92 0d 00 12 dd 31 dd 92  60
0e09 : 20 20 20 36 34 20 4b 42  fe
0e11 : 49 54 20 20 28 38 20 4b  ec
0e19 : 42 29 0d 00 12 ad c3 bd  4c
0e21 : 92 0d 00 12 b0 c3 ae 92  85
0e29 : 0d 00 12 dd 32 dd 92 20  13
0e31 : 20 31 32 38 20 4b 42 49  75
0e39 : 54 20 28 31 36 20 4b 42  e4
0e41 : 29 0d 00 12 ad c3 bd 92  48
0e49 : 0d 00 11 11 1d 1d 1d 1d  26
0e51 : 1d 1d 1d 1d 45 4e 54 45  8a
0e59 : 52 20 44 49 53 4b 20 43  8c
0e61 : 4f 4d 4d 41 4e 44 3a 0d  dc
0e69 : 0d 1d 1d 3d 3e 00 11 9d  57
0e71 : 9d 3c 20 44 45 56 49 43  70
0e79 : 45 20 38 20 4e 4f 54 20  d1
0e81 : 50 52 45 53 45 4e 54 20  0e
0e89 : 3e 91 91 0d 00 11 20 50  40
0e91 : 52 45 53 53 20 12 3c 52  ed
0e99 : 45 54 55 52 4e 3e 92 20  09
0ea1 : 54 4f 20 43 4f 4e 54 49  59
0ea9 : 4e 55 45 00 93 11 20 b0  97
0eb1 : c3 c3 c3 c3 c3 c3 c3 c3  b0
0eb9 : c3 c3 c3 c3 c3 c3 c3 c3  b8
0ec1 : c3 c3 c3 c3 c3 c3 c3 c3  c0
0ec9 : c3 c3 c3 c3 c3 c3 c3 c3  c8
0ed1 : c3 c3 c3 c3 ae 0d 20 dd  6f
0ed9 : 42 59 54 45 53 20 46 52  7a
0ee1 : 45 45 3a 20 20 20 20 20  1f
0ee9 : 20 20 20 42 4c 4f 43 4b  4d
0ef1 : 53 20 46 52 45 45 3a 20  d8
0ef9 : 20 20 20 20 dd 0d 20 dd  b8
0f01 : 20 20 20 20 20 20 20 20  01
0f09 : 20 20 20 20 20 20 20 20  09
0f11 : 20 20 20 20 20 20 20 20  11
0f19 : 20 20 20 20 20 20 20 20  19
0f21 : 20 20 20 20 dd 0d 20 dd  e0
0f29 : 20 20 20 20 20 20 20 20  29
0f31 : 20 20 20 46 49 4c 45 28  8f
0f39 : 53 29 20 4c 4f 41 44 45  4d
0f41 : 44 20 20 20 20 20 20 20  65
0f49 : 20 20 20 20 dd 0d 20 ad  a8
0f51 : c3 c3 c3 c3 c3 c3 c3 c3  50
0f59 : c3 c3 c3 c3 c3 c3 c3 c3  58
0f61 : c3 c3 c3 c3 c3 c3 c3 c3  60
0f69 : c3 c3 c3 c3 c3 c3 c3 c3  68
0f71 : c3 c3 c3 c3 bd 13 11 11  5a
0f79 : 1d 1d 1d 1d 1d 1d 1d 1d  79
0f81 : 1d 1d 1d 1d 1d 1d 00 93  fa
0f89 : 11 9e 8e 08 20 a5 b4 b5  fb
0f91 : a1 12 b6 aa a7 20 92 20  44
0f99 : 4d 4f 44 55 4c 2d 47 45  1f
0fa1 : 4e 45 52 41 54 4f 52 20  98
0fa9 : 56 33 2e 30 20 12 20 a5  89
0fb1 : b4 b5 a1 92 b6 aa a7 0d  74
0fb9 : 0d 9c 20 20 20 20 20 20  e4
0fc1 : 28 43 29 20 31 39 38 35  01
0fc9 : 20 42 59 20 4e 4f 52 42  92
0fd1 : 45 52 54 20 4a 55 4e 47  70
0fd9 : 4d 41 4e 4e 9e 0d 0d 0d  c5
0fe1 : 00 12 b0 c3 c3 ae 0d 00  75
0fe9 : 12 dd 46 31 dd 92 20 20  d5
0ff1 : 44 49 52 45 43 54 4f 52  d0
0ff9 : 59 0d 00 12 ad c3 c3 bd  9f
1001 : 0d 00 12 b0 c3 c3 ae 0d  d8
1009 : 00 12 dd 46 33 dd 92 20  ff
1011 : 20 44 49 53 4b 20 43 4f  71
1019 : 4d 4d 41 4e 44 53 0d 00  3a
1021 : 12 ad c3 c3 bd 0d 00 12  db
1029 : b0 c3 c3 ae 0d 00 12 dd  56
1031 : 46 35 dd 92 20 20 4c 4f  ae
1039 : 41 44 20 46 49 4c 45 0d  93
1041 : 00 12 ad c3 c3 bd 0d 00  8c
1049 : 12 b0 c3 c3 ae 0d 00 12  94
1051 : dd 46 37 dd 92 20 20 53  2c
1059 : 41 56 45 20 4d 4f 44 55  26
1061 : 4c 0d 00 12 ad c3 c3 bd  fa
1069 : 0d 00 12 b0 c3 c3 ae 0d  40
1071 : 00 12 dd 46 38 dd 92 20  b7
1079 : 20 51 55 49 54 0d 00 12  92
1081 : ad c3 c3 bd 0d 00 0d 1d  f8
1089 : 1d 1d 1d 1d 1d 1d 1d 53  f6
1091 : 54 41 52 54 20 20 20 20  69
1099 : 3a 20 52 55 4e 20 3f 20  46
10a1 : 28 59 2f 4e 29 00 0d 0d  ec
10a9 : 1d 1d 1d 1d 1d 1d 1d 1d  a9
10b1 : 53 54 41 52 54 20 20 20  d0
10b9 : 20 3a 20 53 59 53 20 00  1a
10c1 : 0d 1d 1d 1d 1d 1d 1d 1d  b1
10c9 : 1d 53 54 41 52 54 20 20  56
10d1 : 20 20 3a 20 53 59 53 20  22
10d9 : 20 20 20 20 20 20 20 3f  17
10e1 : a0 28 59 2f 4e 29 9d 9d  b1
10e9 : 9d 9d 9d 9d 9d 9d 9d 9d  e8
10f1 : 9d 9d 9d 9d 00 0d 11 1d  5f
10f9 : 1d 1d 1d 1d 1d 1d 1d 46  4c
1101 : 49 4c 45 20 53 41 56 45  e9
1109 : 44 00 49 4e 53 55 46 46  ef
1111 : 49 43 49 45 4e 54 20 4d  99
```

Listing »Modulgenerator«. Bitte beachten Sie die Eingabehinweise auf Seite 7





```
1119 : 45 4d 4f 52 59 00 0d 11  0f
1121 : 1d 57 41 52 4e 49 4e 47  7b
1129 : 3a 20 43 48 41 4e 47 45  7b
1131 : 53 20 4e 4f 54 20 53 41  28
1139 : 56 45 44 21 20 51 55 49  dc
1141 : 54 20 28 59 2f 4e 29 00  e5
1149 : 93 11 11 11 11 11 1d 1d  14
1151 : 1d 1d 1d 51 55 49 54 20  a0
1159 : 54 4f 20 12 20 42 20 92  59
1161 : 41 53 49 43 20 4f 52 20  0d
1169 : 54 45 53 54 20 12 20 4d  6d
1171 : 20 92 4f 44 55 4c 20 3f  ed
1179 : 00 93 0d 11 11 11 11 11  a8
1181 : 11 11 11 11 1d 1d 45 4e  ee
1189 : 54 45 52 20 46 49 4c 45  83
1191 : 4e 41 4d 45 20 3a 0d 1d  be
1199 : 1d 1d 1d 1d 1d 1d 1d 1d  99
11a1 : 1d 1d 1d 1d 1d 1d 1d 1d  a1
11a9 : 1d 1d 2d 2d 2d 2d 2d 2d  91
11b1 : 2d 2d 2d 2d 2d 2d 2d 2d  b1
11b9 : 2d 2d 0d 91 91 1d 1d 1d  a3
11c1 : 1d 1d 1d 1d 1d 1d 1d 1d  c1
11c9 : 1d 1d 1d 1d 1d 1d 1d 1d  c9
11d1 : 00 32 80 5e fe c3 c2 cd  8b
11d9 : 38 30 93 20 2a 2a 20 2a  db
11e1 : 2a 2a 2a 2a 2a 2a 2a 2a  e1
11e9 : 2a 2a 2a 2a 2a 2a 2a 2a  e9
11f1 : 2a 20 4d 4f 44 55 4c 20  c9
11f9 : 56 45 52 53 49 4f 4e 20  79
1201 : 2a 2a 20 78 a2 ff 9a d8  9e
1209 : 8e 16 d0 20 a3 fd 20 50  26
1211 : fd 20 15 fd 20 5b ff 58  b1
1219 : 20 53 e4 20 bf e3 a0 00  be
1221 : a9 80 84 fb 85 fc b1 fb  aa
1229 : 91 fb c8 d0 f9 e6 fc a5  1a
1231 : fc c9 c0 d0 f1 ea ea ea  54
1239 : a9 00 a2 80 85 37 86 38  38
1241 : 20 44 a6 a2 fb 9a a9 00  bc
1249 : 85 fb a9 30 85 fc a2 00  07
1251 : bd 09 80 20 d2 ff e8 e0  49
1259 : 28 d0 f5 e6 fc a2 00 bd  a4
1261 : 57 81 e0 17 d0 02 a5 fc  41
1269 : c9 00 f0 06 20 d2 ff e8  9a
1271 : d0 ed a5 fb 0a 0a 0a 0a  4e
1279 : aa a0 00 bd 81 81 c9 00  76
1281 : f0 09 20 d2 ff e8 c8 c0  44
1289 : 10 30 f0 a2 00 bd 74 81  04
1291 : c9 00 f0 06 20 d2 ff e8  c2
1299 : d0 f3 e6 fb ad 56 81 c5  bb
12a1 : fb d0 b8 e6 fc 20 e4 ff  74
12a9 : f0 fb c5 fc 10 f7 c9 31  f2
12b1 : 30 f3 e9 31 0a 0a 0a 48  25
12b9 : aa bd d1 81 85 5f e8 bd  59
12c1 : d1 81 85 60 e8 bd d1 81  87
12c9 : 85 5a e8 bd d1 81 85 5b  63
12d1 : e8 bd d1 81 85 58 e8 bd  77
12d9 : d1 81 85 59 e8 bd d1 81  be
12e1 : 85 61 e8 bd d1 81 85 62  0d
12e9 : 20 bf a3 20 44 e5 a5 61  a2
12f1 : c9 00 f0 08 c9 01 f0 16  8c
12f9 : 68 6c 61 00 a5 62 c9 00  84
1301 : d0 f6 4c 75 80 20 8e a6  9f
1309 : 20 60 a6 4c ae a7 a5 62  10
1311 : c9 08 d0 e4 68 aa e8 e8  00
1319 : e8 e8 bd d1 81 85 2d e8  ea
1321 : bd d1 81 85 2e d0 de 00  bd
1329 : 0d 1d 1d 1d 1d 1d 1d 1d  19
1331 : 1d b0 c3 ae 0d 1d 1d 1d  d5
1339 : 1d 1d 1d 1d 1d dd 12 1d  13
1341 : 92 dd 20 20 00 0d 1d 1d  e5
1349 : 1d 1d 1d 1d 1d 1d ad c3  d9
1351 : bd 00 00 00 00 00 00 00  0f
1359 : 00 00 00 00 00 00 00 00  5a
1361 : 00 00 00 00 00 00 00 00  62
1369 : 00 00 00 00 00 00 00 00  6a
1371 : 00 00 00 00 00 00 00 00  72
1379 : 00 00 00 00 00 00 00 00  7a
1381 : 00 00 00 00 00 00 00 00  82
1389 : 00 00 00 00 00 00 00 00  8a
1391 : 00 00 00 00 00 00 00 00  92
1399 : 00 00 00 00 00 00 00 00  9a
13a1 : 00 00 00 00 00 00 00 00  a2
13a9 : 00 00 00 00 00 00 00 00  aa
13b1 : 00 00 00 00 00 00 00 00  b2
13b9 : 00 00 00 00 00 00 00 00  ba
13c1 : 00 00 00 00 00 00 00 00  c2
13c9 : 00 00 a2 00 bd 0e 82 f0  aa
13d1 : 06 20 d2 ff e8 d0 f5 ad  e4
13d9 : ff bf c9 e0 d0 f9 60 4d  3f
13e1 : 4f 44 55 4c 20 41 42 53  ed
13e9 : 43 48 41 4c 54 45 4e 00  d3
13f1 : a9 36 85 01 38 a5 5a e5  1d
13f9 : 5f 85 22 a8 a5 5b e5 60  46
1401 : aa e8 98 f0 23 a5 5a 38  9d
1409 : e5 22 85 5a b0 03 c6 5b  a1
1411 : 38 a5 58 e5 22 85 58 b0  00
1419 : 08 c6 59 90 04 b1 5a 91  47
1421 : 58 88 d0 f9 b1 5a 91 58  15
1429 : c6 5b c6 59 ca d0 f2 a9  cc
1431 : 37 85 01 60 00 29 21 ca  db
```

Die besten Tips und Tricks

Es ist einfach lästig, immer wieder -zig Hefte nach einem bestimmten Trick durchsuchen zu müssen. Deshalb haben wir die wichtigsten und interessantesten Tips und Tricks aus früheren 64'er-Ausgaben für Sie zusammengestellt.

Kennen Sie folgende Situation? Man sitzt vor einem Programmierproblem und erinnert sich – da stand doch mal was bei den Tips und Tricks. Ein Stapel 64'er wird geholt, und die Sucherei geht los. Wir wollen da Abhilfe schaffen. Auf den folgenden Seiten finden Sie unter anderem kurze Programme zu den Themen: Basic-Programme retten und verbinden, Bildschirm speichern, Cursor steuern, Listschutz, Maschinenprogramme speichern, PET-Simulation, Reset-Hilfen.

Basic-Programme verbinden

So manch einen C 64-Besitzer wird es schon geärgert haben, daß sein Computer keinen MERGE-Befehl besitzt. Mit wenig Aufwand ist es aber dennoch möglich, Basic-Programme aneinanderzuhängen:

1. Im Direktmodus »PRINT PEEK(43); PEEK(44)« eingeben und sich die Ergebnisse merken.
2. Das erste Programm normal laden.
3. Erscheint jetzt nach »PRINT PEEK(45)« eine Null oder eine Eins, dann geben Sie »POKE 43, 256 + PEEK(45) - 2 : POKE 44, PEEK(46) - 1 : NEW« ein. Im anderen Fall wird »POKE 43, PEEK(45) - 2 : POKE 44, PEEK(46) : NEW« eingegeben.
4. Nun wird das anzuhängende Programm geladen (Achtung! Das anzuhängende Programm muß die höheren Zeilennummern haben).
5. Jetzt POKEn Sie in die Speicherstellen 43 und 44 die zu Anfang gemerkten Werte.

Beide Programme sind nun verbunden. Wichtig bei der ganzen Prozedur ist, daß keine Variablen definiert werden, da das MERGEn sonst nicht richtig funktioniert.

(Thomas Lopatic)

Basic-Programme retten

Die Betriebssystemroutine »Angleich von Koppeladressen« ab Adresse 42291 ermöglicht ein schnelles und einfaches »UNNEW« nach einem versehentlichen »NEW« oder Reset:

```
POKE 2049,1 : POKE 2050,1 : SYS 42291
```

Danach kann zumindest wieder geLISTet werden. Ein vollständiges »UNNEW« verlangt allerdings die Korrektur der Zeiger auf den Beginn der Variablen und Felder. Dazu wäre die Kenntnis der Programmlänge notwendig. Man kann sich aber behelfen, indem man das Programm notfalls in Teilen auf dem Bildschirm auf LISTet und die einzelnen Zeilen mit der RETURN-Taste neu übernimmt.

(Gerhard Wagner)

UNNEW

Haben Sie ein Programm versehentlich mit NEW gelöscht, hilft UNNEW:

```
100 FOR I=525 TO 578
120 READ A : POKE I,A : NEXT I
200 POKE 43,525 AND 255 : POKE 44,2
210 POKE 45,578 AND 255 : POKE 46,2
220 CLR : SAVE " UNNEW",8 : REM bzw. ,1,1
300 DATA 160,003,200,177,043,208,251,200
310 DATA 200,152,160,000,145,043,165,044
320 DATA 200,145,043,133,060,160,000,132
330 DATA 059,162,000,200,208,002,230,060
340 DATA 177,059,208,245,232,224,003,208
350 DATA 242,200,208,002,230,060,132,045
360 DATA 164,060,132,046,096,256
```

Wenn Sie dieses Programm eingeben und starten, wird ein Programm namens »UNNEW« auf Diskette geschrieben. Falls Sie aus Versehen NEW eingetippt haben, dann laden Sie das Programm durch

```
LOAD "UNNEW" ,8,1
```

und starten es durch

```
SYS 525
```

Bildschirm auf Kassette/Diskette

Das kurze Maschinenprogramm (Listing 1) dient dazu, den Bildschirminhalt beim Commodore 64 zu speichern und auch wieder einzuladen. Dies kann mit einem Recorder oder Diskettenlaufwerk geschehen. Dabei werden außer dem eigentlichen Bildschirmspeicher (der an eine beliebige Stelle verschoben sein kann und nicht ab Adresse 1024 liegen muß) auch das Color-RAM und die Tabelle der Doppelzeilenkennzeichnungen mitberücksichtigt. Das Speichern und Laden erfolgt über einen Pufferbereich im »versteckten« RAM ab 40960, da die drei verschiedenen Speicherbereiche in einem Stück gespeichert und außerdem die 1000 Farbnibbles zu 500 Bytes zusammengeschoben werden. Das erspart beim Arbeiten mit dem Recorder wertvolle Zeit.

Nach dem Eintippen des Basic-Loaders kann durch »RUN 16« geprüft werden, ob es richtig eingegeben wurde. Trotzdem sollte es vor dem Start gespeichert werden, da auch durch Prüfsummen nicht alle Fehler erkannt werden können. Ausgegeben werden zwei Adressen für das Laden und Speichern. Das Programm kann an jede Stelle des Arbeitsspeichers geladen werden, in der vorliegenden Version lädt es sich an das Ende des Basic-Arbeitsspeichers. Es ist davon auszugehen, daß das Programm direkt nach dem Einschalten ohne andere Erweiterungen geladen und gestartet wird.

Soll ein Bildschirminhalt auf Diskette gespeichert werden, so erfolgt dies durch

```
SYS 40738 "FILENAME", 8
```

Beim Arbeiten mit Recorder kann (auch beim Laden) die Gerätenummer entfallen; die Angabe einer Sekundäradresse ist nicht erlaubt. Durch das Speichern bedingte Betriebssystemmeldungen wie »PRESS RECORD & PLAY« sowie das Scrolling des Bildschirms sind unbedeutend und werden nicht berücksichtigt. Alle Bildschirmdaten werden zuvor in einen Zwischenspeicher übertragen.

Das Laden solcherart gespeicherter Bildschirminhalte geschieht mit

```
SYS 40833 "FILENAME", 8
```

Zunächst werden die Daten in den Puffer von Adresse 40960 bis 42483 geladen und dann in MSB-Tabelle, Farbspeicher und die momentan gewählte Videomatrix übertragen.

(Ralph Babel)

```
0 REM SCREENSAVER 64                       <057>
1 D=55:A=PEEK(56)*256+PEEK(D)-222:GOSUB 4:
  CLR:S=PEEK(56)*256+PEEK(55)              <216>
2 FOR I=S TO S+221:READ A:POKE I,A:NEXT:A=
  S+173:D=S+1:GOSUB 4:D=S+96:GOSUB 4       <101>
3 PRINT"{DOWN}SAVE:"S:PRINT"LOAD:"S+95:END <255>
4 POKE D,A-INT(A/256)*256:POKE D+1,A/256:R
  ETURN                                    <071>
5 DATA 32,,,162,4,177,7,145,9,200,208,249,
  230,8,230,10,202,208,242,162,24,181      <234>
6 DATA 217,157,231,163,202,208,248,162,218
  ,202,202,134,8,177,7,41,15,133,11,232    <117>
7 DATA 232,134,8,177,7,10,10,10,10,5,11,14
  5,9,200,208,229,230,10,224,219,208,224   <230>
8 DATA 169,160,133,8,165,1,72,41,254,133,1
  ,169,7,162,244,160,165,32,216,255,170    <038>
9 DATA 104,133,1,138,176,1,96,76,249,224,3
  2,,,169,,32,213,255,176,243,165,1,72     <130>
10 DATA 41,254,133,1,162,4,160,,177,9,145,
   7,200,208,249,230,8,230,10,202,208,242  <154>
11 DATA 162,24,189,231,163,149,217,202,208
   ,248,162,218,202,202,134,8,177,9,145,7  <100>
12 DATA 232,232,134,8,74,74,74,74,145,7,20
   0,208,235,230,10,224,219,208,230,104    <131>
13 DATA 133,1,96,32,87,226,162,1,32,121,,2
   40,3,32,241,183,160,2,32,186,255,173,   <036>
14 DATA 221,73,3,133,8,173,24,208,41,240,7
   0,8,106,70,8,106,133,8,160,,132,7,132   <047>
15 DATA 9,169,160,133,10,96                <051>
16 CLR:FOR Q=.TO 221:READ A:X=A+X:Y=A-Y:NE
   XT                                      <057>
17 PRINT"{DOWN}CHECKSUM "MID$("ERROR OK",1
   -5*(X=27349 AND Y=-217),5):END          <027>

© 64'er
```

Listing 1. Listing zu »Bildschirm auf Kassette/Diskette«

Cursorsteuerung leichtgemacht

Bei professionellen Programmen der PC-Klasse kann der Cursor meist über Eingabegeräte wie die Maus positioniert werden. Daß es auch recht gut mit dem Joystick und dem C 64 funktioniert, beweist dieses Programm.

Haben Sie den kurzen MSE-Lader (Listing 2) eingetippt und gestartet, können Sie den Cursor mit einem Joystick in Port 2 steuern. Das Steuerprogramm befindet sich von Adresse \$C000 bis \$C066 im Speicher. Die ersten 15 Bytes nimmt eine Initialisierungsroutine in Anspruch, die den Inter-

```
PROGRAMM : CURSOR              C000 C066
-----------------------------------------

C000 : A9 0F 8D 14 03 A9 C0 8D   B2
C008 : 15 03 A9 06 85 02 60 C6   41
C010 : 02 F0 03 4C 31 EA A9 06   F2
C018 : 85 02 A6 C6 E0 08 90 03   B7
C020 : 4C 31 EA AD 00 DC 29 01   03
C028 : C9 00 D0 05 A9 91 20 5E   2A
C030 : C0 AD 00 DC 29 02 C9 00   2C
C038 : D0 05 A9 11 20 5E C0 AD   6B
C040 : 00 DC 29 04 C9 00 D0 05   63
C048 : A9 9D 20 5E C0 AD 00 DC   C7
C050 : 29 08 C9 00 D0 05 A9 1D   06
C058 : 20 5E C0 4C 31 EA A6 C6   F4
C060 : 9D 77 02 E6 C6 60 00 A0   C7
```

Listing 2. Listing zu »Cursorsteuerung leichtgemacht«. Das Programm muß mit dem MSE eingegeben werden.

```
10 REM**********************************
20 REM*     CURSOR STEUEREUNG              *
30 REM*                                    *
65 REM*        PETER SIEPEN                *
70 REM*                                    *
75 REM*      VON-STEPHAN-STR.6             *
80 REM*                                    *
82 REM*      4200 OBERHAUSEN 1             *
84 REM*                                    *
85 REM*  TELEFON :(0208)/26555             *
90 REM**********************************
95 :
100 SYS9*4096
110 .OPT P,OO
111 :
112 :
113 :
114 :
120 *= $C000
125 :
130 :
140 :
150 TEST  = $02          ;ZAEHLVARIABLE
160 JOY   = 56320        ;PORT #2
170 AZITP = $C6          ;ANZAHL ZEICHEN IM PUFFER
174 :
175 :
180 LDA #<BEGINN;INTERUPTVEKTOR
185 STA $314
190 LDA #>BEGINN;AUF NEUE
195 STA $315     ;ADDRESSE SETZEN
200 LDA #$06     ;ZAEHLVARIABLE
205 STA TEST     ;HOCHSETZEN
210 RTS          ;ZURUECK ZU BASIC
215 :
220 :
225 :
300 BEGINN   DEC TEST
320 :        BEQ START    ;GENUG LEER IRR
330 :        JMP $EA31    ;NEIN WEITER MIT IRR
340 START    LDA #$06     ;ZAEHLVARIABLE HOCHSETZEN
350 :        STA TEST
360 :        LDX AZITP    ;TASTATURPUFFER
370 :        CPX #$08     ;VOLL
380 :        BCC WEITER
390 :        JMP $EA31    ;JA WEITER MIT INTERUPT
400 WEITER   LDA JOY      ;WENN JOY NICHT
410 :        AND #1       ;NACH OBEN
420 :        CMP #0       ;WEITER
430 :        BNE NOBEN
440 :        LDA #"□"     ;WENN JA STEUERZEICHEN
450 :        JSR AUSG     ;AUSGEBEN
460 NOBEN    LDA JOY
470 :        AND #2
480 :        CMP #0
490 :        BNE NUNTEN
500 :        LDA #"▣"
510 :        JSR AUSG
520 NUNTEN   LDA JOY
530 :        AND #4
540 :        CMP #0
550 :        BNE NLINKS
560 :        LDA #"▮"
570 :        JSR AUSG
580 NLINKS   LDA JOY
590 :        AND #8
600 :        CMP #0
700 :        BNE NRECHTS
710 :        LDA #"◨"
720 :        JSR AUSG
730 NRECHTS  JMP $EA31
740 :
750 :
760 :
770 :
780 :
790 :
800 AUSG     LDX $C6      ;X-REG FUER
810 :        STA $277,X   ;INDIZIERTE
820 :        INC AZITP    ;ADDRESSIERUNG
830 :        RTS          ;LADEN
840 :                     ;IN TASTPUFFER
850 :                     ;SCHREIBEN
860 :                     ;ANZAHL ZEICHEN
870 :                     ;IM TASTPUFFER
880 :                     ;ERHOEHEN
890 :
900 :
READY.
```

Listing 3. Assemblerprogramm der Cursorsteuerung

ruptvektor auf $COOF legt und das Steuerprogramm in den Kernelinterrupt einbindet. Die Routine wird mit SYS 49152 aufgerufen. Das Programm benutzt Speicherzelle $02 als Zählregister, da der Joystick nur bei jedem sechsten Interrupt abgefragt wird. In Zeile 300 des Assemblerlistings (Listing 3) wird die Zählvariable um 1 erniedrigt. Ist die Variable 0, wird sie auf 6 gesetzt und in die Steuerungsroutine verzweigt. Ab Zeile 360 wird der Tastaturpuffer auf freien Platz überprüft. Sollte der Puffer voll sein, wird sofort in die Interruptroutine des Betriebssystems ($EA31) gesprungen. In Zeile 400 wird der Joystick abgefragt und das entsprechende Cursorsteuerzeichen in den Akku geladen. Ab 800 wird das Zeichen in den Tastaturpuffer geschrieben und der Pufferzeiger erhöht.

(P. Siepen)

Cursor steuern

Das Betriebssystem des C 64 enthält eine Routine, die man benutzen kann, um den Cursor an eine beliebige Stelle zu setzen. Geben Sie doch mal folgendes ein:

```
POKE 214, (Zeile) : POKE 211, (Spalte) : SYS 58640 :
PRINT " TEXT"
```

(Michael Keukert)

Cursor beschleunigt

Für alle diejenigen C 64-Besitzer, denen die Bewegung des Cursors bisher zu langsam war, gibt es einen speziellen POKE.

Mit `POKE 56325,5` wird der Cursor rasend schnell und flitzt bei Betätigung der Cursortasten nur noch so über den Bildschirm. Wer's lieber gemütlicher mag, der sollte es statt dessen einmal mit `POKE 56325,255` probieren.

(Oliver Bausch)

Directory ohne Programmverlust

Häufig möchte man sich das Directory einer Diskette ansehen, ohne das gerade im Speicher befindliche Programm zu zerstören. Wenn man das DOS 5.1 nicht geladen hat, behilft man sich meist mit der zeitaufwendigen Zwischenspeicherung des Programms auf der Diskette. Es geht jedoch auch einfacher und schneller. Geben Sie einfach den folgenden Befehl ein:

```
POKE 44, PEEK(46) + 1
```

Damit wird der Basic-Anfang auf einen freien Speicherbereich gestellt. Sie können jetzt wie gewohnt mit »`LOAD "$",8`« das Directory laden und anschließend auflisten.

Mit `POKE 44,8` sind Sie dann wieder im eigentlichen Programm.

(Heinzpeter Oelkers)

Floppy-Lister

Mit einem einfachen SYS-Aufruf können Sie Programme und sequentielle Dateien direkt von Diskette listen. Programme im Speicher bleiben dabei erhalten.

»Floppy-Lister« bietet zwei Möglichkeiten, ein Programm von Diskette zu listen. Basic-Programme und sequentielle Dateien können entweder als Klartext oder als Speicherauszug (Dumps) gelistet werden. Im Dumpmodus werden sämtliche Daten als Hexcodes ausgegeben und, soweit möglich, in ASCII-Zeichen übersetzt. Deshalb eignet er sich besonders

64ER ONLINE

zum Analysieren von unbekannten Programmdateien. Maschinenprogramme dürfen grundsätzlich nur auf diese Art gelistet werden, will man einen Absturz vermeiden.

Ist das Ladeprogramm (Listing 4) mit dem MSE eingetippt, kann das Maschinenprogramm auf Diskette oder Kassette gespeichert werden. Der »Floppy-Lister« kann dann direkt geladen (`LOAD "name", 8, 1`, oder `LOAD "name", 1, 1`) und mit `SYS 49152,"XY:filename"` aufgerufen werden. Vergessen Sie nach dem absoluten Laden nicht, den erforderlichen NEU-Befehl im Direktmodus einzugeben, um die Basic-Zeiger in einen vernünftigen Zustand zu bringen. »X« steht für die Fileart: entweder »P« für Programm oder »S« für sequentielle Datei. »Y« steht für den Modus, in dem gelistet werden soll: »L« für einfaches Listen und »D« für Ausgabe in Form eines Speicherauszugs. Zum Beispiel listet der Befehl `SYS 49152,"SL:TESTSEQUENZ"` eine sequentielle Datei mit dem Namen »TESTSEQUENZ«. Abkürzungen des Filenamens mit »*« sind erlaubt. Falsche Eingaben werden mit einer Fehlermeldung quittiert.

Mit der CTRL-Taste kann die Ausgabe verlangsamt und mit der RUN/STOP-Taste unterbrochen werden. Eine Fortsetzung erfolgt mit der A-, und vorzeitiger Abbruch des Listens mit der DEL-Taste. Am Ende eines Listings muß immer die SPACE-Taste gedrückt werden.

Das Maschinenprogramm für »Floppy-Lister« liegt im Bereich von \$C000 -\$C2B2 (49152-49842). An das Programm schließt sich noch ein Pufferbereich an, wo Daten zwischengespeichert werden. "Floppy-Lister" läuft mit den gängigen Erweiterungen, wie DOS 5.1, Simons Basic und Exbasic.

(B. Schulzki)

```
programm : floppylister      c000 c2b5
--------------------------------------

c000 : 20 fd ae 20 9e ad 20 a3  ee
c008 : b6 a5 64 85 a7 a5 65 85  a3
c010 : a8 a0 00 b1 a7 99 1f c3  8a
c018 : c8 c0 03 90 f6 ad 1f c3  f4
c020 : c9 14 b0 1e c9 04 90 24  2a
c028 : a0 00 ad 21 c3 85 a8 ad  be
c030 : 20 c3 85 a7 b1 a7 99 22  8b
c038 : c3 c8 cc 1f c3 d0 f5 4c  a9
c040 : 53 c0 a9 6f a0 c2 20 1e  28
c048 : ab 4c 74 a4 a9 88 a0 c2  b2
c050 : 4c 46 c0 a2 00 bd 22 c3  42
c058 : 8d 1e c3 c9 50 f0 08 c9  5f
c060 : 53 d0 e9 a0 04 d0 02 a0  ba
c068 : 00 e8 bd 22 c3 c9 4c f0  2e
c070 : 06 c9 44 d0 d7 c8 c8 e8  3f
c078 : bd 22 c3 c9 3a d0 cd 8c  eb
c080 : 0e c3 ae 1f c3 a9 2c 48  ca
c088 : 9d 22 c3 e8 ad 1e c3 9d  5a
c090 : 22 c3 68 e8 9d 22 c3 a9  18
c098 : 52 e8 9d 22 c3 a9 00 e8  65
c0a0 : 9d 22 c3 ee 1f c3 ad 28  34
c0a8 : 03 8d 0f c3 ad 29 03 8d  f9
c0b0 : 10 c3 a9 22 8d 28 03 a9  ca
c0b8 : c2 8d 29 03 ad 1f c3 a2  14
c0c0 : 25 a0 c3 20 bd ff a9 02  b1
c0c8 : a2 08 a0 02 20 ba ff 20  ef
c0d0 : c0 ff a2 02 20 c6 ff ac  0a
c0d8 : 0e c3 b9 ab c2 8d 11 c3  10
c0e0 : b9 ac c2 8d 12 c3 6c 11  65
c0e8 : c3 a9 00 8d 13 c3 8d 14  df
c0f0 : c3 8d 15 c3 a2 00 20 0f  00
c0f8 : c2 b0 03 4c e9 c1 20 cf  29
c100 : ff 9d 16 c3 a5 90 d0 05  f8
c108 : e8 e0 08 90 e9 8d 15 c3  5b
c110 : a9 00 e8 e0 08 b0 07 9d  6d
c118 : 16 c3 e8 4c 13 c1 20 4c  2c
c120 : c2 ad 14 c3 20 2a c2 ad  f0
c128 : 13 c3 20 2a c2 a9 3a 20  0d
c130 : d2 ff a2 00 bd 16 c3 20  86
c138 : 23 c2 e8 e0 08 90 f5 20  30
c140 : 4c c2 a9 12 20 d2 ff a2  78
c148 : 00 bd 16 c3 29 7f c9 20  1b
c150 : b0 04 a9 2e d0 03 bd 16  7b
c158 : c3 20 d2 ff e8 e0 08 90  b7
c160 : e8 a9 0d 20 d2 ff ad 15  72
c168 : c3 d0 11 18 ad 13 c3 69  30
c170 : 08 8d 13 c3 90 03 ee 14  81
c178 : c3 4c f4 c0 4c e9 c1 20  12
c180 : 51 c2 20 51 c2 a2 ff 20  e6
c188 : 51 c2 a5 a7 e8 9d b3 c2  68
c190 : e0 03 90 f3 20 51 c2 e8  fe
c198 : a5 a7 9d b3 c2 d0 f5 e8  4b
c1a0 : 9d b3 c2 e8 9d b3 c2 a9  ba
c1a8 : b3 85 5f a0 c2 84 60 20  1c
c1b0 : 37 a5 ad 00 03 48 ad 01  50
c1b8 : 03 48 a9 22 8d 00 03 a9  c6
c1c0 : c2 8d 01 03 20 c3 a6 68  75
c1c8 : 8d 01 03 68 8d 00 03 20  c9
c1d0 : 0f c2 b0 b1 4c e9 c1 20  fe
c1d8 : cf ff 20 d2 ff a5 90 d0  1a
c1e0 : 08 20 0f c2 b0 f1 4c e9  b4
c1e8 : c1 20 cc ff a9 02 20 c3  9f
c1f0 : ff ad 0f c3 8d 28 03 ad  84
c1f8 : 10 c3 8d 29 03 a9 60 a0  b3
c200 : c2 20 1e ab a5 cb c9 3c  28
c208 : d0 fa a9 00 85 c6 60 a5  1b
c210 : cb f0 0e c9 3f d0 08 a5  f6
c218 : cb f0 06 c9 0a d0 f8 38  92
c220 : 60 18 60 48 a9 20 20 d2  6f
c228 : ff 68 48 4a 4a 4a 4a 20  17
c230 : 42 c2 a8 68 29 0f 20 42  1b
c238 : c2 48 98 20 d2 ff 68 4c  b0
c240 : d2 ff 18 69 f6 90 02 69  14
c248 : 06 69 3a 60 a9 20 4c d2  10
c250 : ff 20 cf ff 85 a7 a5 90  a0
c258 : f0 05 68 68 4c e9 c1 60  ce
c260 : 0d 20 20 20 20 12 20 53  43
c268 : 50 41 43 45 20 0d 00 0d  57
c270 : 46 45 48 4c 45 52 3a 20  05
c278 : 4e 41 4d 45 20 5a 55 20  cd
c280 : 4c 41 4e 47 20 21 0d 00  29
c288 : 0d 46 45 48 4c 45 52 3a  bf
c290 : 20 46 41 4c 53 43 48 45  a8
c298 : 53 20 45 49 4e 47 41 42  1f
c2a0 : 45 46 4f 52 4d 41 54 20  97
c2a8 : 21 0d 00 7f c1 e9 c0 d7  5e
c2b0 : c1 e9 c0 32 a9 00 ff 00  77
```

Listing 4. »Floppy-Lister«

Joystickabfrage:

Auch die Joystickabfrage ist beim C64 nicht ganz einfach. Dabei hilft folgendes Hilfsprogramm:

```
10 POKE 56322,224
20 JO=PEEK(56320)
30 IF (JO AND 1)=0 THENPRINT"OBEN"
40 IF (JO AND 2)=0 THENPRINT"UNTEN"
50 IF (JO AND 4)=0 THENPRINT"LINKS"
60 IF (JO AND 8)=0 THENPRINT"RECHTS"
70 IF (JO AND 16)=0 THENPRINT"FEUER"
```

Dieses Programm gilt für eine Joystickabfrage am Controlport 2.

Commodore-Joystick verbessert

Der Commodore-Joystick VIC-1311 benötigt eine relativ große Hebelbewegung, um die Kontakte zu schließen. Bei Spielen, die eine hohe Reaktionsgeschwindigkeit erfordern, ist diese Eigenschaft sehr ungünstig. Man kann jedoch recht einfach Abhilfe schaffen:

Man entfernt die vier Schauben an der Unterseite des Gehäuses und hebt den oberen Teil des Joysticks mit der Platine vorsichtig ab. Nun wird die Platine an den Durchtrittsöffnungen der Schrauben mit je einer etwa 1 Millimeter dicken Unterlegscheibe verstärkt. Die vier Unterlegscheiben können mit einem Tropfen Alleskleber (Vorsicht, nicht die Kontakte verkleben!) gegen Verrutschen gesichert werden. Danach wird der Joystick wieder zusammengeschraubt. Wenn Sie alles richtig gemacht haben und insbesondere keine Teile übriggeblieben sind, dann werden die Kontakte des Joysticks nun bei erheblich kleineren Hebelbewegungen geschlossen.

(M. Kunze)

64'er online

64er-online.de

List-Schutz für Basic-Programme

Dieser List-Schutz (Listing 5) ist für Nichteingeweihte sehr verblüffend. Die Grundidee dazu stammt aus dem Bericht »Disketten-Manipulationen« aus der 64'er, Ausgabe 6/85. Er wurde jedoch etwas ausgebaut, so daß hier beim Listen alle Steuercodes aktiv werden. Dies wird dadurch erreicht, daß man in eine Speicherstelle vor den Codes die Zahl 141 schreibt. Um nun ein Programm zu schützen, lädt man es und gibt zwei Zeilen ein (Listing 5).

Danach gibt man im Direktmodus ein:

```
POKE 2067,71 : POKE 2073,141 : POKE 2118,0 :
POKE 2119,0
```

Listet man nun das Programm, so wird der Bildschirm gelöscht und der Text in der REM-Zeile ausgegeben. Durch das künstlich erzeugte Basic-Programm-Endezeichen (drei Nullen) wird das Listen abgebrochen. Wird im Programm dann auch noch durch `POKE 788,52` die RUN-STOP-, und durch `POKE 792,193` die RESTORE-Taste ausgeschaltet, kann keiner mehr an das Programm. Aufheben läßt sich dieser List-Schutz nur mit einem Monitor und mit der Kenntnis der Funktionsweise des Schutzes.

(Thomas Uttendorfer)

```
1 POKE 2067,73:GOTO 10                          <134>
2 REM"A{CLR,3DOWN,YELLOW}IT IS NOT ALLOWED
   TO LIST THIS PROGRAM{BLUE}AA                 <123>
```

© 64'er

Listing 5. Listing zu »List-Schutz für Basic-Programme«

List- und Löschschutz leichtgemacht

Es wurden schon viele Methoden veröffentlicht, um ein Basic-Programm gegen Listen zu schützen. Aber alle mir bekannten Möglichkeiten weisen entschiedene Nachteile auf. Entweder der Schutz ist nicht sicher genug und leicht zu entfernen, oder er ist viel zu aufwendig.

Ich habe mich daher entschlossen, ein Programm zu schreiben, das diese Mängel umgeht und sogar noch andere positive Merkmale aufweist.

Zunächst eine Zusammenfassung von drei mir bekannten Listschutzmöglichkeiten mit ihren Vor- und Nachteilen:

1. Möglichkeit

In die erste Zeile des Basic-Programms (zum Beispiel Zeilennummer 1) wird REM, gefolgt von zwei Anführungszeichen und SHIFT L, geschrieben.

```
1 REM""L (RETURN)
```

Der Cursor wird nun auf das zweite Anführungszeichen gesetzt und sechsmal SHIFT INST gedrückt (das Anführungszeichen wird um sechs Positionen nach rechts geschoben). Dann wird sechsmal DEL eingegeben (es erscheinen als Steuerzeichen sechs reverse T) und die Zeile mit (RETURN) abgespeichert. Wird nun der LIST-Befehl aufgerufen, meldet sich der Rechner mit:

"

?SYNTAX ERROR
READY.

Auf den ersten Blick sehr beeindruckend, aber durch Entfernen dieser Zeile ist der Listschutz wieder aufgehoben. Außerdem ist ein »LIST 2« noch möglich.

2. Möglichkeit

In jede Basic-Zeile werden synthetische Steuerzeichen eingefügt (genaue Beschreibung im 64'er-Magazin, Ausgabe 6/84). Diese Methode ist zwar recht sicher, will man aber alle Zeilen eines längeren Basic-Programms schützen, ist der Aufwand viel zu groß, vom Speicherplatzbedarf der Steuerzeichen ganz abgesehen.

3. Möglichkeit

Durch `POKE 775,200` wird der Listbefehl außer Kraft gesetzt, durch `POKE 775,167` wird diese Wirkung wieder aufgehoben. Dieser Listschutz ist zwar wirkungsvoll, aber er muß erst durch diesen POKE-Befehl aktiviert werden. Nach dem Laden eines Programms ist er daher noch nicht aktiv.

Das hier vorgestellte Programm erzeugt nicht nur einen sicheren Listschutz, sondern schützt auch vor dem Löschen einzelner Basic-Zeilen. So können zum Beispiel Hinweise auf ein Kopierrecht und auf den Autor eines Programms nicht geändert oder entfernt werden. Auch kann ein so gesichertes Programm nur mit RUN gestartet werden, ein RUN, gefolgt von einer Zeilennummer, führt zu einer Fehlermeldung. Jede Zeile des Programms ist geschützt, es können also auch einzelne Zeilen nicht gelistet werden. Einzige Bedingung für die Verwendung dieses Schutzes: Das zu schützende Programm darf keine Zeilennummern 0 und 1 enthalten. Ansonsten wird eine Fehlermeldung ausgegeben und das Programm bleibt unverändert.

```
0 REM **********************************     <131>
1 REM *       (C) U. V. GAISBERG       *     <195>
2 REM *        AM ZUCKERBERG 70        *     <080>
3 REM *        7140 LUDWIGSBURG        *     <233>
4 REM *        TEL. 07141/55910        *     <056>
5 REM **********************************     <136>
6 FOR I=0 TO 340:READ A:B=B+A:POKE 50000+I
  ,A                                         <099>
7 NEXT I                                     <091>
8 IF B <> 33527 THEN PRINT"FEHLER IN DATAS
   !":END                                    <042>
9 PRINT"OK !":END                            <113>
10 REM DATAS FUER MASCHINENPROGRAMM          <157>
11 DATA 169,0,141,32,208,141,33,208,169,1,
   141,134,2,32,68,229,174,3,8,172           <018>
12 DATA 4,8,192,0,208,7,224,2,176,3,76,206
   ,195,162,0,142,134,2,169,32,32            <221>
13 DATA 210,255,232,224,50,208,246,162,0,1
   89,21,196,157,0,4,232,224,29,208          <001>
14 DATA 245,169,24,157,0,4,232,224,69,208,
   246,162,0,189,50,196,157,80,4,232         <190>
15 DATA 224,8,208,245,162,0,189,58,196,157
   ,120,4,232,224,8,208,245,162,10           <073>
16 DATA 160,0,24,32,240,255,169,19,162,13,
   160,4,141,119,2,142,120,2,142,121         <061>
17 DATA 2,142,122,2,132,198,96,162,0,189,9
   9,196,32,210,255,232,224,31,208           <065>
18 DATA 245,96,32,68,229,162,10,160,0,24,3
   2,240,255,162,1,142,134,2,202,189         <210>
19 DATA 66,196,32,210,255,232,224,33,208,2
   45,169,20,162,17,160,255,141,18           <250>
20 DATA 8,142,29,8,140,4,8,162,0,189,130,1
   96,157,32,8,232,224,34,208,245            <171>
21 DATA 96,48,18,5,13,34,148,148,148,148,1
   48,148,148,148,148,148,34,12,12           <122>
22 DATA 9,19,20,19,3,8,21,20,26,26,76,49,1
   9,25,19,50,48,57,56,19,25,19,53           <045>
23 DATA 48,49,52,48,80,82,79,71,82,65,77,7
   7,32,45,32,40,67,41,32,85,46,86           <224>
24 DATA 46,71,65,73,83,66,69,82,71,32,32,4
   9,57,56,52,66,73,84,84,69,32,90           <032>
25 DATA 69,73,76,69,32,48,32,85,78,68,32,4
   9,32,69,78,84,70,69,82,78,69,78           <005>
26 DATA 32,33,169,255,141,4,8,169,131,162,
   164,141,2,3,142,3,3,76,131,164            <172>
27 DATA 165,2,141,4,8,169,32,162,8,141,2,3
   ,142,3,3,96,0                             <150>
```

© 64'er

Listing 6. Listing zu »List- und Löschschutz leichtgemacht«

Das Listschutzprogramm (Listing 6) liegt als Basic-Lader vor. Nachdem es richtig abgetippt wurde, kann es durch RUN gestartet werden. Das Maschinenprogramm steht dann im Speicher ab der Adresse 50000 zur Verfügung. Das zu schützende Basic-Programm kann nun geladen werden, durch SYS 50000 wird das Schutzprogramm aktiviert und das Basic-Programm geschützt. Es kann nun wieder auf Kassette/Diskette gespeichert werden. Das mit dem Listschutz versehene Programm ist nur um wenige Bytes größer als vorher.

Funktionsweise

Das Maschinenprogramm (Listing 6) generiert zwei Basic-Zeilen mit den Zeilennummern 0 und 1. Die Zeile 0 ist eine REM-Zeile, in der ein unlistbares Zeichen (SHIFT L) steht. Hinter diesem Zeichen stehen dann noch zwei kurze Maschinenprogramme, deren Funktionen im folgenden noch erklärt werden. In der zweiten Zeile steht ein SYS-Befehl, der eine der beiden Maschinenroutinen in Zeile 0 startet. Sind diese beiden Zeilen nun erzeugt, wird die Zeilennummer 0 durch eine höhere, eigentlich unerlaubte Zeilennummer (größer 64000) ersetzt. Diese Zeile kann daher auch nicht gelöscht werden.

Da alle nun folgenden Zeilen des Programms kleiner sind als die erste, können diese vom Computer nicht mehr erkannt werden. Ein Sprung in eine solche Zeile führt zu der Fehlermeldung: ?UNDEF'D STATEMENT ERROR. Es kann daher auch keine Zeile gelöscht werden, da diese für den Computer ja nicht mehr vorhanden sind.

Der einzige Nachteil ist, daß es nicht nur ein perfekter List- und Löschschutz, sondern auch ein RUN-Schutz ist (auch Sprungziele innerhalb des Programms können nicht gefunden werden).

Wird das geschützte Programm gestartet, trifft der Interpreter als erstes auf den SYS-Befehl in Zeile 1. Es folgt ein Sprung in das Maschinenprogramm in der REM-Zeile. Dort wird die Zeilennummer wieder auf 0 gesetzt, und der Vektor auf den Basic-Warmstart wird auf die zweite Maschinenroutine gesetzt.

Nun kann das Basic-Programm ohne Fehler ausgeführt werden. Wird der Programmlauf unterbrochen (durch STOP-Taste, Fehlermeldungen, Programmende und so weiter), wird das zweite Maschinenprogramm über den Basic-Warmstartvektor angesprungen. Dort wird die Zeilennummer wieder hochgesetzt, der Warmstartvektor wieder auf den normalen Wert gebracht und die Warmstartroutine angesprungen. Das Programm liegt nun wieder in der geschützten Form vor.

(Ulrich von Gaisberg)

Maschinenprogramme auf Tastendruck

Mit einer kleinen Routine kann man ein Maschinenprogramm mit einem Tastendruck aufrufen. Dafür benutzt man ein Zeichen, das normalerweise nicht oder nur selten verwendet wird. Ich habe mich für das @-Zeichen entschieden.

Im Interpreter existiert eine Schleife, die einen Basic-Befehl holt und ausführt.

```
A7E1 JMP (0308) ; zeigt normalerweise auf A7E4
A7E4 JSR 0073   ; nächstes Zeichen aus Basic-Text holen
A7E7 JSR A7ED   ; Statement ausführen
A7EA JMP A7AE   ; zurück zur Interpreterschleife
```

In den Speicherzellen $0308 und $0309 (776 und 777 dez) liegt ein Zeiger, der normalerweise auf den Beginn dieser Schleife zeigt. Verbiegt man nun den Zeiger auf eine eigene Routine, kann man den Basic-Befehl auf das eigene Zeichen überprüfen.

Wird es erkannt, springt man auf den Anfang des gewünschten Unterprogramms. Wurde das Zeichen nicht vorgefunden, macht man in der Schleife normal weiter.

Dieses Verfahren verwende ich bei der Programmierhilfe »Merge« aus Ausgabe 4/84, die normalerweise mit SYS 50000 gestartet werden muß. Es kann aber auch für andere Maschinenprogramme umgeschrieben werden.

»Merge« belegt den Speicherbereich 50000 bis 50264. Die eigene Routine beginnt auf Adresse 49152 (C000 hex). Der Wert in den Adressen $0308 und $0309 muß deshalb auf C000 abgeändert werden. Der Computer durchläuft dann vor jedem Befehl, den er ausführen soll, folgende Schleife:

```
C000 JSR 0073 ; nächstes Zeichen holen
C003 CMP   40 ; Vergleich mit @-Zeichen
C005 BEQ      ; verzweigen wenn erkannt
C007 JSR 0079 ; Flags setzen
C00A JMP A7E7 ; Rücksprung
C00D JSR 0073 ; nächstes Zeichen holen
C010 JSR C350 ; zur eigenen Routine
C013 JMP A7AE ; Rücksprung
```

Nach dem Drücken von @ und RETURN wird nun das Programm ab Adresse 50000 (C350 Hex) ausgeführt. Auf die anderen Befehle hat diese Routine keinen Einfluß. Eine Hürde gibt es noch zu meistern. Die Änderung der Adressen 0308 und 0309 ist auf der Basic-Ebene mit POKE nicht möglich. Dies ist auch verständlich, da POKE auch ein Basic-Befehl ist und durch die Änderung der ersten Adresse der Einsprung verändert wird.

Deshalb muß diese Adreßänderung in Maschinensprache durchgeführt werden.

```
C100 LDA   00 ; Lade Akku mit 00
C102 STA 0308 ; Speichere Akku nach 0308
C105 [illegible]   C0 ; Lade Akku mit C0
C107 STA 0309 ; Speichere Akku nach 0309
C10A RTS      ; Rückkehr nach Basic
```

Basic-Lader für Befehlserweiterung

```
240   FOR I= 49152 TO 49152 + 21
250   READ Q : POKE I, Q
260   NEXT
300   FOR I = 49408 TO 49408 + 10
310   READ Q : POKE I, Q
320   NEXT : SYS 49408
11000 DATA 32,115,0,201,64,240,6,32,121
12000 DATA 0,76,231,167,32,115,0,32,80
13000 DATA 195,76,174,167
14000 DATA 169,0,141,8,3,169,192,141,9,3,96
```

Diesen Basic-Lader tippt man hinter das Programm »Merge« und speichert es gemeinsam ab.

Die Zeilennummern sind so gewählt, daß man sie direkt zum Basic-Lader von »Merge« dazutippen kann. In Zeile 10260 müssen aber dann die letzten fünf Nullen gelöscht werden.

Mit SYS 49408 wird die Befehlserweiterung aktiviert und steht dann zur Benutzung bereit.

(Patrik Fleig)

Maschinenprogramme auf Diskette speichern

Reine Maschinenprogramme haben gegenüber einem Basic-Lader zwei entscheidende Vorteile. Sie sind wesentlich kürzer und können direkt, ohne zeitraubendes POKEn, geladen werden. Dieses Programm soll Ihnen helfen, Maschinenprogramme auf Diskette zu speichern.

Ein einfacher Basic-Lader besteht aus einer READ/POKE-Schleife, in der DATAs in den Speicher geschrieben werden.

64er-online.de
64er-online.net

```
50000 REM    BASIC-LADER                          <034>
50010 DATA   ..................                   <058>
50020 DATA   ..................                   <068>
50030 DATA   ..................                   <078>
50040 :                                           <231>
50050 REM    MASCHINENPROGRAMME AUS               <065>
50060 REM    DATALADERN                           <219>
50070 :                                           <005>
50080 REM S KOENNEN WEGGELASSEN WERDEN            <056>
50090 :                                           <025>
50100 REM   DA = STARTADRESSE                     <249>
50110 REM   EA = ENDADRESSE                       <125>
50120 :                                           <055>
50130 OPEN 1,8,1, "NAME"                          <131>
50140 :                                           <075>
50150 REM SA U.EA ZUORDNEN                        <036>
50160 :                                           <095>
50170 SH = INT(SA/256) :REM HIGH-BYTE             <166>
50180 SL = SA-SH*256   :REM LOW-BYTE              <064>
50190 PRINT#1,CHR$(SL);CHR$(SH);                  <065>
50200 :                                           <137>
50210 FOR I=SA TO EA                              <249>
50220 READ WERT                                   <064>
50230 PRINT#1,CHR$(WERT);                         <079>
50240 NEXT I                                      <029>
50250 :                                           <187>
50260 CLOSE 1                                     <232>
```

© 64'er

Listing 7. Listing zu »Maschinenprogramme auf Diskette speichern«

Jeder Wert, der in dieser Schleife gePOKEt wird, ist ein Byte eines Maschinenprogramms, ein Maschinenbefehl oder ein Teil davon. Anstatt die DATAs in den Speicher zu POKEn, kann ein Basic-Lader dazu benutzt werden, das Maschinenprogramm auf Diskette abzuspeichern. Der Lader wird dazu nur etwas geändert.

Ein Maschinenprogrammfile auf Diskette enthält, vor den Maschinenbefehlen, in den ersten beiden Bytes, die Startadresse des Programms. Der Computer erkennt daran, ab welcher Speicherstelle er das Programm laden soll.

Der Basic-Lader (Listing 7) muß also ein Programmfile eröffnen, die Startadresse des Maschinenprogramms ins File schreiben und anschließend alle Werte aus den DATA-Zeilen. Wie das im einzelnen programmiert wird, können Sie am Listing des Beispielprogramms sehen, das Sie ohne weiteres an jedes Ladeprogramm anpassen können.

Der wichtigste Befehl in diesem Beipiel steht in Zeile 50130:

```
OPEN 1,8,1, "Programmname"
```

Er öffnet eine Datei. Allerdings keine sequentielle, sondern eine Programmdatei. Die Sekundäradresse 1 sagt der Floppy 1541, daß nun ein Programm (NAME.PRG) geschrieben wird. Häufig findet man für diesen Befehl die Syntax

```
OPEN 2,8,2, "Programmname,P, W"
```

bei der man die Sekundäradresse frei wählen darf. Hier muß jedoch nach dem Namen angegeben werden, daß eine Programmdatei (P) geöffnet werden soll, in die geschrieben wird (Write). Entsprechend kann bei sequentiellen Dateien optional »S,W« bei Schreib- oder »S,R« bei Lesezugriffen angegeben werden.

Nachdem in Zeile 50130 die Programmdatei angelegt wurde, muß die Startadresse des Maschinenprogrammes gespeichert werden. Die Startadresse wird dazu in den Zeilen 50170 und 50180 in Low- und High-Byte zerlegt und die zwei Byte in Zeile 50190 in das Programmfile geschrieben. Jetzt wird das eigentliche Programm gespeichert. Die numerischen Werte müssen dazu mit der CHR$-Funktion in die entsprechenden Strings übersetzt werden. Um die erforderliche Datendichte beim Schreiben des Files zu erreichen, muß jedem String ein Semicolon »;« folgen. Ohne Semicolon ist das Programm später nicht lauffähig. Sind alle Werte, die der Basic-Lader in den Speicher gePOKEt hätte, gespeichert, wird die Datei mit CLOSE geschlossen. Damit ist das Maschinenprogramm auf Diskette gespeichert und kann mit

```
LOAD "NAME",8,1
```

geladen und, wenn dies notwendig ist, mit dem entsprechenden SYS-Befehl gestartet werden.

Laden von Maschinenprogrammen in Basic

Sie können Maschinenroutinen mit Hilfe von Basic-Programmen laden, ohne sie zu zerstören. Dabei ist zu beachten, daß nach dem Laden das Basic-Programm von neuem startet. Die Basic-Zeile

```
10 LOAD "KEIN ENDE",8,1
```

bewirkt deshalb eine Endlosschleife, die immer wieder das Maschinenprogramm »KEIN ENDE« lädt. Da angelegte Variablen dabei erhalten bleiben, kann mit

```
10 IF A=0 THEN A=1:LOAD "KEIN ENDE",8,1
```

ein mehrmaliges Laden verhindert werden, da beim Neustart A=1 ist.

(S. Wengler)

Der C 64 als PET

Wenn Sie CBM 2000, 3000 oder 4000 geschriebene Programme auf Ihrem C 64 laufen lassen wollen, müssen Sie umständlich PEEKs und POKEs ändern. Der »Pet-Simulator« nimmt Ihnen diese Arbeit ab.

Ist das Programm (Listing 8) eingegeben und gestartet, werden als erstes die DATAs für das Maschinenprogramm in den Bereich ab Adresse 49152 gePOKEt (SU = Prüfsumme für die Daten). Danach fragt das Programm nach der Zeichenfarbe. Sie werden aufgefordert, eine Zahl zwischen 0 und 15 einzugeben.

(0 = schwarz 1 = weiß,..., 15 = grau 3).

```
1 REM ***********************************   <132>
2 REM *        PET - SIMULATOR          *   <251>
3 REM *                                 *   <052>
4 REM *      (C) BY W. HOPF 1984        *   <218>
5 REM ***********************************   <136>
6 :                                         <238>
7 REM PROGRAMM VOR DEM START ABSPEICHERN     <071>
8 :                                         <240>
10 FOR I=49152 TO 49152+91:READ A:SU=SU+A   <231>
20 POKE I,A:NEXT                            <117>
30 IF SU<>12552 THEN END                    <064>
40 PRINT"{CLR,2DOWN}BITTE WAEHLEN SIE DIE
   ZEICHENFARBE"                            <063>
50 PRINT"{DOWN}(0-15 EINGEBEN)!{DOWN}"      <124>
60 INPUT ZF:IF ZF>=0 AND ZF<=15 THEN POKE
   49239,ZF:POKE 646,ZF:GOTO 80             <089>
70 PRINT"{DOWN}NICHT ERLAUBT":FOR I=1 TO 1
   000:NEXT:GOTO 40                         <066>
80 SYS 49152:POKE 1,54:PRINT"{CLR}PET-SIMU
   LATOR AKTIV":NEW                         <208>
10000 DATA 160,0,132,254,169,160,133,255    <121>
10005 DATA 177,254,145,254,200,208,249,230  <103>
10010 DATA 255,166,255,224,192,208,241      <189>
10015 DATA 169,5,141,0,221,141,24,208,169   <221>
10020 DATA 128,141,136,2,133,56,169,4       <229>
10025 DATA 133,44,169,0,141,0,4,169         <241>
10030 DATA 63,141,37,184,169,192,141,38     <018>
10035 DATA 184,169,193,141,24,3,96,32       <211>
10040 DATA 235,183,24,165,21,201,128,144    <095>
10045 DATA 18,201,132,176,14,24,105,88      <177>
10050 DATA 133,255,165,20,133,254,169,0     <209>
10055 DATA 234,145,254,96                   <037>
60000 :                                     <029>
60010 REM LISTE DER VERWENDETEN COMMODORE-
      STEUERZEICHEN                         <140>
60020 REM "{CLR}" = CLR                     <245>
60030 REM "{DOWN}" = CRSR-DOWN              <186>
```

Listing 8. Listing zu »Der C 64 als PET«

Bei anschließendem Starten des Maschinenprogrammes wird das Basic-ROM in das darunterliegende RAM gePOKEt (Basic-Interpreter kopieren). Anschließend wird das Bildschirm-RAM von Adresse 1024 nach Adresse 32768 verlegt. Basic-Speicheranfang und -ende werden dem des PET angepaßt. Weiterhin wird in der POKE-Routine des Basic-Interpreters ein Eingriff vorgenommen, nach der der Computer aus dem Interpreter in eine Routine des Maschinenprogramms springt. Hier wird überprüft, ob das Bildschirm-RAM angesprochen wurde. Trifft dies zu, wird die dazugehörige Farb-RAM-Adresse berechnet und der vorher festgelegte Farbwert (Zeichenfarbe) hineingePOKEt. Um das Zurücksetzen des Bildschirmes auf das C 64-Format zu vermeiden (durch Drücken der RUN/STOP- und RESTORE-Tasten), wird die RESTORE-Taste durch Verändern des NMI-Vektors ausgeschaltet. Programme können aber noch mit der RUN/STOP-Taste unterbrochen werden.

Nach Ablauf des Maschinenprogrammes meldet sich der Computer mit »PET-SIMULATOR AKTIV«. Sie können jetzt immer noch die Zeichenfarbe mit POKE 49239, ZF (ZF = Zeichenfarbe – siehe oben) ändern. Wenn Sie jetzt zum Beispiel POKE 32768,1 eingeben, erscheint ein »A« am linken oberen Bildschirmrand in der gewählten Zeichenfarbe. Schlußbemerkung: Bevor Sie das Programm starten, empfiehlt es sich, es vorher zu speichern, da sich das Programm selbständig löscht.

(Wolfgang Hopf)

Pseudo-Interrupt

Diese Befehlserweiterung erlaubt es, ein Basic-Programm zu jedem beliebigen Zeitpunkt per Tastendruck durch die F1-Taste unterbrechen zu lassen. Es kann dann in eine vorher definierte Basic-Routine gesprungen werden. Diese könnte zum Beispiel den noch freien Speicherplatz oder die Uhrzeit anzeigen.

Das Programm (Listing 9) bitte mit dem MSE eingeben. Es liegt dann im Speicherbereich von 40499 bis 40768. Geladen wird es absolut mit `LOAD "PSEUDO-IRQ",8,1`. Da das Programm im Bereich für die Basic-Variablen steht, muß es durch `POKE 56,158:CLR` vor Überschreiben geschützt werden. Nach dem Start durch »`SYS 40541`« stehen die neuen Befehle zur Verfügung:

!F1JUMP <Zeilennummer> legt fest, in welche Zeile im Falle einer Unterbrechung durch die F1-Taste gesprungen werden soll. Tritt der Befehl mehrmals auf, so gilt die zuletzt angegebene Zeilennummer.

!JBACK bewirkt die Fortsetzung des Basic-Programms ab der Stelle, an der unterbrochen wurde.

!SF1 verhindert Unterbrechungen. Dies kann zum Beispiel beim Aufbau einer Grafik oder bei Arbeiten mit der Diskettenstation wichtig sein.

!CF1 läßt gesperrte Unterbrechungen wieder zu.

(Guido Schuhmacher)

```
programm : pseudo-irq        9e01 9f41
-----------------------------------------

9e01 : ff 89 00 20 73 00 c9 21  6a
9e09 : d0 03 4c 6d 9e a5 cb c9  f5
9e11 : 40 d0 06 20 79 00 4c e7  d8
9e19 : a7 c9 04 f0 06 20 79 00  0b
9e21 : 4c e7 a7 a5 cb c9 40 d0  ad
9e29 : fa ad 01 9e f0 06 20 79  c1
9e31 : 00 4c e7 a7 18 a9 03 20  61
9e39 : fb a3 a5 7a 48 a5 7b 48  ef
9e41 : a5 39 48 a5 3a 48 a9 75  c1
9e49 : 48 8d 01 9e ad 02 9e 85  dc
9e51 : 14 ad 03 9e 85 15 20 a3  99
9e59 : a8 4c ae a7 a9 04 8d 08  c9
9e61 : 03 a9 9e 8d 09 03 a9 75  cc
9e69 : 8d 01 9e 60 20 73 00 c9  5c
9e71 : 46 f0 07 c9 4a f0 41 4c  f4
9e79 : f1 9e 20 73 00 c9 31 d0  e5
9e81 : 34 20 73 00 c9 4a d0 2d  2f
9e89 : 20 73 00 c9 55 d0 26 20  51
9e91 : 73 00 c9 4d d0 1f 20 73  8e
9e99 : 00 c9 50 d0 18 20 73 00  fc
9ea1 : 20 6b a9 a5 14 8d 02 9e  89
9ea9 : a5 15 8d 03 9e a9 00 8d  ef
9eb1 : 01 9e 4c ae a7 4c 08 af  47
9eb9 : 20 73 00 c9 42 d0 f6 20  93
9ec1 : 73 00 c9 41 d0 ef 20 73  c3
9ec9 : 00 c9 43 d0 e8 20 73 00  f6
9ed1 : c9 4b d0 e1 68 c9 75 d0  fc
9ed9 : 59 68 85 3a 68 85 39 68  77
9ee1 : 85 7b 68 85 7a a9 00 8d  ff
9ee9 : 01 9e 20 79 00 4c e7 a7  c2
9ef1 : c9 53 f0 07 c9 43 f0 1c  34
9ef9 : 4c 08 af 20 73 00 c9 46  24
9f01 : d0 b3 20 73 00 c9 31 d0  d6
9f09 : ac a9 75 8d 01 9e 20 73  05
9f11 : 00 4c ae a7 20 73 00 c9  09
9f19 : 46 d0 9a 20 73 00 c9 31  33
9f21 : d0 93 a9 00 8d 01 9e 20  c1
9f29 : 73 00 4c ae a7 4a 42 41  de
9f31 : 43 cb a9 2e 85 22 a9 9f  d9
9f39 : 85 23 4c 47 a4 01 00 a5  e9
```

Listing 9. Listing zu »Pseudo-Interrupt«

Es muß nicht immer »READY.« sein...

Wenn Sie sich darüber ärgern, daß der C 64 nach jedem ausgeführten Befehl sein stupides »READY.« auf den Bildschirm schreibt, dann geben Sie doch die beiden folgenden Zeilen im Direktmodus ein:

```
FOR I=40960 TO 49151 : POKE I,PEEK(I) : NEXT :
POKE 1,54 : REM Basic-Interpreter ins RAM laden
FOR I= 41848 TO 41853 : POKE I,32 : NEXT :
REM READY-Meldung überschreiben
```

Wenn Sie jetzt irgendeinen Befehl eingeben, erscheint kein »READY.« mehr. Natürlich können Sie aber auch einfach den Text ändern:

```
A$="HALLO." : FOR I=1 TO 6 : POKE 41847+I,ASC
(MID$(A$,I,1)) : NEXT
```

Nach Eingabe dieser Zeile meldet sich der Interpreter nach jeder Eingabe mit »Hallo.«. Sie können jeden beliebigen Text wählen, vorausgesetzt, er ist maximal sechs Zeichen lang.

(Andreas Scharrer)

Reset

Falls Sie in einem Programm einen Reset wünschen, dann benutzen Sie folgenden Befehl:

```
SYS 64738
```

Mit folgendem kleinen Programm sorgt man von Basic aus dafür, daß ein Reset keinen Effekt hat:

```
100 FOR I=32770 TO 32778
110 READ A : POKE I,A
120 NEXT I
130 DATA 10, 128, 195, 194, 205
140 DATA 56, 48, 88, 0
```

(Daniel Kossmann)

Einfacher Reset-Schalter selbstgebaut

Sehr preiswert und einfach kommt man im Selbstbau zu einem Reset-Schalter, wenn man sich im Elektronikfachgeschäft einen sechspoligen Diodenstecker und einen Miniatur-Tastschalter besorgt. Der Tastschalter wird zwischen die Pins 2 und 6 des Diodensteckers gelötet. Zur Pinbelegung vergleiche auch Seite 142 im C 64-Handbuch. Der

64er-online.de

so präparierte Diodenstecker wird nun in den seriellen Port des C 64 beziehungsweise VC 20 gesteckt. Falls der Port bereits von Floppy oder Drucker belegt ist, kann der Stecker natürlich auch in die freie Buchse des entsprechenden Zusatzgerätes eingesteckt werden. Auf Knopfdruck wird nun in jedem Fall wieder der Einschaltzustand des Computers hergestellt.

Die Kosten für Stecker und Taste betragen etwa 4 Mark.

(Henning Zipf)

Reset-Helfer für C 64

Das Betriebssystem des C 64 enthält ab der Adresse $FD02 ein Unterprogramm, das im Steckmodulbereich ab $8000 nach der Zeichenfolge »CBM80« sucht. Nach dem Einschalten des Computers oder nach einem Reset wird dieses Unterprogramm jedesmal aufgerufen. Werden ab der Adresse $8003 die Zeichen »CBM80« gefunden, dann wird nicht zum Basic-Start gesprungen, sondern das Betriebssystem nimmt an, daß ein Modul eingesteckt ist, holt sich aus der Speicherzelle $8000/$8001 die Startadresse des Modulprogramms und verzweigt dorthin.

Das kleine »Reset-Helfer«-Programm (Listing 10) nutzt dies aus, indem es die genannten Speicherstellen in geeigneter Weise abändert. Es schreibt die »CBM80«-Zeichenfolge ab $8003 in RAM und läßt die Speicherstellen $8000/$8001 auf den Basic-Warmstart zeigen.

Wenn man jetzt einen Reset auslöst, kommt man wie gewohnt aus allen »abgestürzten« Programmen heraus, ein vorhandenes Basic-Programm bleibt aber erhalten.

(Henning Zipf)

RESTORE für Unterprogramme

Will man in ein Programm ein schon vorhandenes Unterprogramm einfügen, kann es beim Lesen von DATAs Schwierigkeiten geben. Oft genügt die RESTORE-Anweisung nicht.

Wenn beide Programmteile, Hauptprogramm und Unterprogramme, DATAs enthalten, muß sichergestellt werden, daß auch wirklich die richtigen Werte gelesen werden. Wenn man nicht aufpaßt, kann es passieren, daß das Unterprogramm DATAs aus dem Hauptprogramm liest. Wie kann man das verhindern? Es gibt eine umständliche Methode: Man kann eine kleine Basic-Erweiterung einbauen, den RESTORE X-Befehl. Es geht aber auch einfacher. Die Zeropage, das sind die ersten 256 Bytes des Speichers, hilft uns bei der Lösung des Problems. Genauer gesagt, die Adressen 65/66 und 122/123. Schlagen wir im C 64-Handbuch auf Seite 162 nach, dann steht dort:

65- 66 Adresse des aktuellen DATA-Elements
122-123 Basic-Zeiger innerhalb der Subroutine

Mit diesen Informationen läßt sich schon etwas anfangen. Wenn das Unterprogramm angesprungen wird, dann sollte der Zeiger in Speicherstelle 122/123 auf die Adresse des Unterprogramms im Speicher stehen. POKEt man diese Werte in die Zeilen 65/55 mit

```
POKE 65,PEEK(122)
POKE 66,PEEK(123),
```

so wird beim nächsten READ der Wert gelesen, der hinter dieser Basic-Zeile mit den POKEs steht, also das erste DATA-Element innerhalb des Unterprogramms. Nach dem Rücksprung aus dem Unterprogramm muß der Zeiger eventuell auch im Hauptprogramm wieder gestellt werden.

In dem kurzen Demo-Listing (Listing 11) werden drei Unterprogramme in zufälliger Reihenfolge aufgerufen.

(Stephan Pätzold)

```
100 REM *** RESET-HELFER ***                 <141>
110 REM                                      <172>
120 REM HENNING ZIPF                         <168>
130 REM KIRCHSTR. 8                          <036>
140 REM 6086 RIEDSTADT 5                     <150>
150 REM TEL. (06158) 72453                   <243>
160 REM                                      <222>
170 FOR I=1 TO 9                             <016>
180 READ D                                   <244>
190 POKE 32767+1,D                           <196>
200 NEXT I                                   <028>
210 POKE 53280,14:POKE 53281,6               <166>
220 PRINT CHR$(147);CHR$(5)                  <071>
230 PRINT" C 64 CHANGED RESET VEKTOR"        <067>
240 PRINT                                    <086>
250 PRINT" 64 K RAM SYSTEM 38911 BASIC BYT
    ES FREE"                                 <233>
260 PRINT                                    <108>
270 NEW                                      <154>
280 DATA 0,0,255,0,195,194,205,56,48         <015>
290 REM ERST SAVE, DANN RUN !                <160>
```

Listing 10. Listing zu »Reset-Helfer für C 64«

RAM-Floppy

Wer kennt das nicht: ein paar Veränderungen an einem Programm – eine Zeile rein, eine andere raus – und nichts geht mehr. Das lästige Neuladen des Originalprogramms von Diskette können Sie ab jetzt vergessen.

Ist ein Programm mal wieder zu Tode editiert, werden Sie nun nicht mehr von den langen Ladezeiten der 1541 in Ihrem

```
1 REM    *************************           <128>
2 REM    *          DEMO         *           <010>
3 REM    *  SUBROUTINE-RESTORE   *           <071>
4 REM    *************************           <131>
5 PRINT"{CLR,6SPACE}TASTE DRUECKEN !"        <104>
6 PRINT:PRINT                                <114>
10 X=INT(RND(TI)*3)+1                        <125>
20 ON X GOSUB 1000,2000,3000                 <040>
25 POKE 65,PEEK(122):POKE 66,PEEK(123)       <001>
30 READ A$:PRINT A$                          <066>
50 DATA " HAUPTPRG."                         <146>
100 GOTO 10                                  <078>
1000 REM   *** SUBROUTINE 1 ***              <183>
1005 :                                       <042>
1010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <221>
1020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <145>
1030 READ A$:PRINT A$;                       <105>
1040 POKE 198,0:WAIT 198,1                   <116>
1050 DATA 1,11,111,1111,"UP1{2SPACE}"        <085>
1060 RETURN                                  <182>
1070 :                                       <108>
2000 REM   *** SUBROUTINE 2 ***              <164>
2005 :                                       <022>
2010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <201>
2020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <125>
2030 READ A$:PRINT A$;                       <084>
2040 POKE 198,0:WAIT 198,1                   <095>
2050 DATA 2,22,222,2222,"UP 2 "              <076>
2060 RETURN                                  <162>
2070 :                                       <088>
3000 REM   *** SUBROUTINE 3 ***              <145>
3005 :                                       <002>
3010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <181>
3020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <105>
3030 READ A$:PRINT A$;                       <064>
3040 POKE 198,0:WAIT 198,1                   <075>
3050 DATA 3,33,333,3333,"UP{2SPACE}3"        <066>
3060 RETURN                                  <141>
3070 :                                       <067>
```

Listing 11. Demoprogramm zu »RESTORE für Unterprogramme« mit zwei POKEs

Programmierdrang gebremst. Mit »RAM-Floppy« (Listing 12) kann ein Programm bearbeitet werden, während man eine Kopie davon im RAM hat. In Sekundenschnelle kann die Kopie in den Basic-Speicher gebracht oder mit der Orginalversion vertauscht werden. Ganz einfach durch Eingabe von »@V« oder »@T«. Numerische Variablen bleiben dabei erhalten. Die »RAM-Floppy« besitzt eine Speicherkapazität von maximal 25 KByte. Der Speicher beginnt ab Adresse 40960.

Ein Problem ergibt sich im Speicherbereich des Kernal- und Basic-ROMs. Ein POKE-Befehl schreibt ins RAM, während die PEEK-Funktion auf das ROM zugreift. Noch komplizierter sieht es beim Zeichen-ROM und den I/O-Bausteinen aus. Wie Sie vielleicht aus unserem Grafikkurs wissen, gibt es in diesen Bereichen drei Speicheretagen. Der Inhalt der Zelle 1 regelt den Zugriff des Computers auf die verschiedenen Speicherebenen. Werden die Bits 0 und 1 in Adresse 1 gelöscht, sieht der Computer nur noch das RAM. Basic- und Kernel-ROM sind verschwunden. Löschen Sie diese Bits deshalb nur durch ein Maschinenspracheprogramm, wenn vorher Ein- und Ausgaben gesperrt wurden. Dies wird durch Setzen des Interruptregisters erreicht.

Das Maschinenprogramm besteht aus drei Teilen. Im Bereich von 40704 bis 40768 erfolgt die Auswertung der Befehle von »RAM-Floppy« und der Aufruf der beiden Unterprogramme, die das Tauschen oder Verschieben der Basic-Programme erledigen.

Das abgedruckte Basic-Programm (Listing 12) POKEt das Maschinenprogramm ab Adresse 40704. Mit `SYS 40704` wird es initialisiert.

Die »RAM-Floppy« hat eine Kapazität von 25 KByte. Das Programm im Basic-Speicher kann zwar 38 KByte lang sein, läßt sich dann allerdings nicht mehr vollständig verschieben oder vertauschen. Der Speicherbedarf sollte auch bei Programmen mit vielen Variablen nicht außer acht gelassen werden. Bei langen Programmen mit vielen Variablen kann es durchaus vorkommen, daß die Programme zwar getauscht, die Variablen allerdings nicht mehr übernommen werden können.

(Uwe Klatt)

```
0 REM***********************************     <037>
1 REM*          RAM-FLOPPY             *     <029>
2 REM***********************************     <039>
3 REM*  UWE KLATT                      *     <042>
4 REM*  BILLERBECKER STR. 27           *     <148>
5 REM*  4939 STEINHEIM                 *     <067>
6 REM*  TEL. 05233/5672                *     <106>
7 REM***********************************     <044>
8 POKE 53280,0:POKE 53281,11                 <021>
9 POKE 646,8                                 <072>
10 PRINT"BITTE WARTEN"                       <040>
11 REM ******************************        <023>
12 REM *** DATAS  LESEN           ***        <020>
13 REM ******************************        <025>
14 FOR I=40704 TO 40768:READ A:POKE I,A:S=
   S+A:NEXT                                  <100>
15 IF S<>6567 THEN END                       <181>
16 FOR I=40784 TO 40849:READ A:POKE I,A:S=
   S+A:NEXT                                  <143>
17 IF S<>14392 THEN END                      <098>
18 FOR I=40853 TO 40902:READ A:POKE I,A:S=
   S+A:NEXT                                  <169>
19 IF S<>20412 THEN END                      <147>
20 REM ******************************        <032>
21 REM ***        MENUE           ***        <197>
22 REM ******************************        <034>
23 PRINT CHR$(147)                           <052>
24 PRINT"{RVSON,40SPACE}";                   <194>
25 PRINT"{RVSON,SPACE}RAM-FLOPPY 25.5 KBYT
   E{18SPACE}";                              <048>
26 PRINT"{RVSON,40SPACE}"                    <246>
27 PRINT"'@V'{2SPACE}VERSCHIEBT PROGRAMM I
   N{2SPACE}RAM-FLOPPY"                      <254>
28 PRINT"'@T'{2SPACE}VERTAUSCHT PROGRAMM M
   IT RAM-FLOPPY"                            <075>
29 REM ******************************        <041>
30 REM *** MC PROGRAMM STARTEN   ***         <222>
31 REM ******************************        <043>
32 SYS 40704:NEW                             <211>
33 REM ******************************        <045>
34 REM *** DATAS FUER 1. MC TEIL ***         <097>
35 REM ******************************        <047>
36 DATA 169,159,133,56,133,52,169,0,133,55
   ,169,21,141,8,3,169                       <063>
37 DATA 159,141,9,3,96,32,115,0,240,4,201,
   64,240,3,76,231                           <111>
38 DATA 167,32,115,0,201,84,240,7,201,86,2
   40,12,76,8,175,32                         <255>
39 DATA 115,0,32,80,159,76,174,167,32,115,
   0,32,149,159,76,174                       <127>
40 DATA 167                                  <008>
41 REM ******************************        <053>
42 REM *** DATAS FUER 2. MC TEIL ***         <234>
43 REM ******************************        <055>
44 DATA 169,0,133,45,169,104,133,46,120,16
   5,1,41,252,133,1,169                      <051>
45 DATA 0,133,98,133,100,141,0,160,169,160
   ,133,101,169,8,133,99                     <246>
46 DATA 162,96,160,0,177,98,133,102,177,10
   0,145,98,165,102,145,100                  <094>
47 DATA 200,208,241,230,99,230,101,202,208
   ,232,165,1,9,3,133,1                      <062>
48 DATA 88,96                                <112>
49 REM ******************************        <061>
50 REM *** DATAS FUER 3. MC TEIL ***         <114>
51 REM ******************************        <063>
52 DATA 120,165,1,41,252,133,1,169,0,133,9
   8,133,100,141,0,160                       <069>
53 DATA 169,160,133,101,169,8,133,99,162,9
   6,160,0,177,98,145,100                    <197>
54 DATA 200,208,249,230,99,230,101,202,208
   ,240,165,1,9,3,133,1                      <197>
55 DATA 88,96                                <119>
```

Listing 12. Listing zu »RAM-Floppy«

Sequentielle Datei als Basic-Programm laden

Es sind eine Reihe von Anwendungen denkbar, bei denen aus einer sequentiellen Datei auf Diskette oder Kassette ein lauffähiges Basic-Programm erstellt werden soll (Datenfernübertragung, Umwandlung von Textfiles in Basic-Programme). Der folgende Einzeiler macht's für den VC 20 möglich:

```
OPEN 1, ⟨Gerät⟩, ⟨Sekundäradresse⟩, "⟨Name⟩" :
POKE 812, 238 : POKE 781, 1 : SYS 65478
```

Dieses Miniprogramm öffnet das File Nummer 1 als Eingabefile (anstelle der Tastatur). Außerdem wird der CLALL-Vektor des Betriebssystems auf ein »RTS« gesetzt, so daß beim Einlesen von Programmzeilen keine Files geschlossen werden.

Deshalb werden nach Eingabe der obigen Befehlszeile von der ausgewählten Datei so lange Zeilen eingelesen und anschließend sofort im Direktmodus ausgeführt, bis die Betriebssystem-Routine CLRCHN aufgerufen wird (zum Beispiel durch einen Syntaxfehler in den gelesenen Zeilen oder durch GET #1, A$).

Um in den normalen Eingabemodus zurückzukehren, muß nur »POKE 812, 239 : CLR« eingegeben werden.

Zum Ausprobieren: Laden Sie ein beliebiges Basic-Programm und geben Sie danach im Direktmodus ein:

```
OPEN 1,8,3,"LISTING,S,W": CMD 1 : LIST : PRINT#1 :
CLOSE 1
```

Spezialeffekt

Wenn man beim C 64 in die Speicherstelle 53270 Werte zwischen 0 und 15 schreibt (POKE 53270,x), kann man den Bildschirm um bis zu sieben Bildpunkte nach links oder rechts scrollen lassen. Ist x kleiner als 8, dann scrollt der Bildschirmausschnitt um x Bildpunkte nach links, sonst um x-8 Bildpunkte nach rechts.

POKE 53270,8 stellt den Normalzustand wieder her.
Dieser Trick läßt sich gut bei Action-Spielen als optische Untermalung beispielsweise einer Explosion einsetzen.

(Michael Keukert)

Text und Grafik mischen

Im Leserforum des 64'er-Magazins, Ausgabe 8/84, fragte Frank Schager nach einer Möglichkeit, mit Simons Basic ein Textfenster in der hochauflösenden Grafik zu erzeugen.
Mit dem folgenden kleinen Programm wird die normale Tastaturabfrage mit dem Simons Basic-Befehl »TEXT« verbunden:

```
10 X = 2 : Y = 2 : REM Text-Anfang
20 HIRES 15, 11 : REM Grafik ein
30 GET A$
40 IF A$ <  >  " " THEN GOSUB 100
50 GOTO 30
60 REM
100 X = X + 8 : REM X-Koordinate erhöhen
110 IF X >38 * 8 THEN X = 2 : Y = Y + 8 :
REM Zeilenende? Dann neue Zeile
120 TEXT X,Y,A$,1,1,8 : REM Zeichen drucken
130 AA$ = AA$ + A$ : REM Wort erzeugen
140 IF AA$ = "GEHE" THEN 1000 : REM Zum Beispiel
150 RETURN
```

Veränderbar ist auch der Faktor 38 in Zeile 110, je nachdem, welche Zeilenlänge gewünscht wird. Ebenso besteht die Möglichkeit, zwischen den Zeilen 130 und 150 weitere IF-Abfragen einzubauen. Empfehlenswert ist die besondere Abfrage der Tasten SPACE und RETURN.

(Jörg Prante)

VC 20-Grundversion simuliert

Mit der folgenden kurzen Routine lassen sich die meisten Grundversions- oder +3 KByte-Programme auch mit einer 8 KByte-Erweiterung laden und ausführen:

```
POKE 648, 30 : SYS 64821
POKE 4096, 0 : POKE 44, 16 : NEW
```

Da der Bildschirmspeicher durch die kleine Routine an der gleichen Stelle wie in der Grundversion liegt, sind die meisten Grundversions-Programme direkt lauffähig.

Automatische Zeilennumerierung

Das lästige Durchnumerieren der Zeilen bei der Programmierung kann Ihnen dieses kleine Programm abnehmen.
Die Syntax des AUTO-Befehls ist:
←A anfangszeilennummer, schrittweite
Nach Eingabe dieses Befehls wird die Zeilennummer vorgegeben und nach RETURN um »schrittweite« erhöht.
Um aus dem AUTO-Modus wieder herauszukommen, muß man nach Vorgabe einer Zeilennummer
»←« RETURN eingeben.
Falls man nach Vorgabe einer Zeilennummer die RETURN-Taste betätigt, wird die entsprechende Zeile, falls sie vorhanden ist, gelöscht. Hiermit lassen sich auch sehr schnell Programmblöcke löschen, falls man die RETURN-Taste gedrückt hält, die Zeilenvorgabe weiterläuft und die entsprechenden Zeilennummern gelöscht werden.
»←«=CHR$(95)
»A«=CHR$(65)
Das Programm (Listing 13) als Basic-Lader eintippen, anschließend mit RUN starten. Falls »FEHLER IN DEN DATAZEILEN« erscheint, DATAs auf Tippfehler überprüfen. Falls »OK«, kann die Basic-Erweiterung mit SYS 49152 initialisiert werden. Nun hat man das Basic um den Befehl »A« erweitert.

(Frank Siedel)

```
1010 REM**********************************   <130>
1020 REM**      AUTO FUER C 64          **   <230>
1030 REM**           VON                **   <129>
1040 REM**      FRANK SIEDEL            **   <145>
1050 REM**      POSENER STR. 18         **   <101>
1060 REM**      2945 SANDE              **   <232>
1070 REM**********************************   <192>
1080 :                                       <040>
1090 :                                       <050>
1100 :                                       <060>
1110 :DATA 169,11,141,8,3,169,192,141,9,3,
     96,32,115,0,8,201,95,240,4,40,76,231   <016>
1120 :DATA 167,32,115,0,201,65,208,245,32,
     115,0,24,32,107,169,165,20,133,38      <106>
1130 :DATA 165,21,133,39,32,253,174,24,32,
     107,169,165,20,133,40,165,21,133,41    <160>
1140 :DATA 169,129,141,2,3,169,192,141,3,3
     ,169,128,141,138,2,165,39,133,98,165   <110>
1150 :DATA 38,133,99,162,144,56,32,73,188,
     32,221,189,162,0,189,1,1,240,9,157     <044>
1160 :DATA 0,2,32,210,255,232,208,242,32,1
     8,225,201,95,240,30,201,13,240,45      <002>
1170 :DATA 157,0,2,232,32,98,165,76,134,16
     4,24,165,38,101,40,133,38,165,39,101   <017>
1180 :DATA 41,133,39,76,75,192,169,131,141
     ,2,3,169,164,141,3,3,169,0,141,138     <181>
1190 :DATA 2,40,76,116,164,32,118,165,76,1
     34,164                                 <239>
1200 :                                       <160>
1210 :                                       <170>
1220 PRINT CHR$(147)                         <233>
1230 SU=0                                    <254>
1240 FOR I=1 TO 170                          <217>
1250 READ A                                  <018>
1260 SU=SU+A                                 <107>
1270 POKE 49151+I,A                          <145>
1280 NEXT                                    <020>
1290 IF SU<>17417 THEN PRINT "FEHLER IN DE
     N DATAZEILEN":END                      <228>
1300 PRINT"OK":END                           <023>
```

© 64'er

Listing 13.
Listing zu »Automatische Zeilennumerierung«

Zwei Tips für den C64

Die Speicherstellen 57 und 58 enthalten die Zeilennummer der aktuellen Basic-Zeile. Die Abfrage geschieht mit »PRINT PEEK(57) + 256 * PEEK(58)«.
Mit »PRINT PEEK(1)« kann abgefragt werden, ob eine Taste an der Datasette gedrückt ist. Es gibt drei mögliche Werte:
7: Taste gedrückt,
55: keine Taste gedrückt,
39: Taste gedrückt, aber Programmlauf unterbrochen.
Diese Abfragen sind für die benutzerfreundliche Programmierung von Kassettenoperationen recht nützlich.

(Wolfgang Meyer/kn)

Und noch ein Tip

Der FORMULAR TOO COMPLEX-Error ist sehr unangenehm, da sich das Programm danach oft nicht mehr listen läßt. Nach Eingabe von POKE 24,0 verhält sich der Computer aber wieder normal.

(Roger Limberg)

Tips und Tricks, die nur ein PEEK oder POKE beinhalten, finden Sie in der POKE-Liste. (kn)

Kurz und nützlich - Einzeiler

Einzeiler sind häufig sehr nützliche Utilities oder »Basic-Erweiterungen«. Wir stellen Ihnen deshalb hier einige dieser Kurzprogramme vor.

Wir wollen uns nicht nur auf's Listing beschränken, sondern Ihnen auch eine Beschreibung der Variablen und eine kurze Erklärung des Programmlaufs geben. So wird es sicher einfacher, die Kniffe, die in solchen Einzeilern stecken, zu verstehen und auch selbst zu verwenden.

Zahlenkonvertierungen von Dezimal nach Hexadezimal

Diese Zahlenkonvertierungen braucht man recht häufig.
- Hex X$ nach dezimal (Listing 1)

```
— Hex X$ nach dezimal X:10
x=0:fori=ltolen(x$):
x0=asc (mid$(x$,i,l)):x=16*x
+x0-48+(x0>64)*7: next
```

und - Dezimal X nach hex X$: (Listing 2)

```
— Dezimal X nach hex X$:10
x$="":fori=lto4:x0=x/16:
x=x-int(x0)*16:x$=chr$
(48+x-(x>9)*7+x$:x=
x0:next
```

Umwandlung beliebiger Zahlensysteme

Die folgenden beiden Einzeiler von Martin und Hartmut Sprave dienen zum Umwandeln von Dezimalzahlen in Zahlen beliebiger Basis und umgekehrt. Man kann die beiden Programme auch kombinieren und erhält so eine Umwandlungsroutine zwischen verschiedenen Zahlensystemen. Beide sind auch als Unterroutine in einem Programm denkbar.

Zur Umwandlung dezimal/beliebig:
Die Variable D enthält eine Dezimalzahl beliebiger Größe. In der Variablen B muß die Basis angegeben werden, die der umgewandelten Zahl zugrunde liegt. Das Ergebnis steht in $Z.

Zum Programm: Die Dummy-Schleife (von 0 bis 0) wird benutzt, um später wieder mitten in die Zeile springen zu können. An jeder Stelle wird die Zahl in der Variablen D durch die Basis B geteilt. Dadurch wird die unterste Ziffer abgeschnitten. Die jeweils niederwertigste Stelle ist der ganzzahlige Rest dieser Division und steht in S. Dieser wird in ASCII-Code umgerechnet. Durch den CHR$-Befehl wird der Code zu einer Zeichenkette. Diese wird vorne an $Z angehängt. Die letzte höchstwertige Ziffer ist erreicht, wenn D<1, da die nächste Stelle dann 0 ist. Solange D>=1 ist, ist die Endbedingung noch nicht erreicht und die FOR-Schleife wird weiter durchlaufen (bis P+1<=0).

```
10 z$="":forp=0to0:d=d/b:s=(d-int(d))*b:
z$=chr$(55+s+7*(s<10))+z$:p=-d:next
40 rem in 10 dez in beliebig
```

Zur Umwandlung beliebig/dezimal:

Diese Routine wandelt eine beliebig große Zahl, deren Basis in der Variablen B steht, in eine Dezimalzahl um. Die Zahl selbst muß in $Z stehen. D wird mit 0 initialisiert. Die Schleifenvariable S dient als Zeiger auf die einzelnen Stellen von $Z. Diese werden nacheinander in ASCII-Code umgewandelt. Der Code für Null (48) wird subtrahiert und das Ergebnis in H zwischengespeichert. Man multipliziert die umgewandelten Stellen (in D) mit B und erhöht sie dadurch um eine Potenz dieser Basis. Dann addiert man die aktuelle Stelle (H) dazu. Bei Darstellung durch einen Buchstaben (H>9) ist aufgrund des ASCII-Codes noch die Subtraktion einer 7 notwendig. Die Schleife wird solange durchlaufen, bis die niederwertigste Stelle erreicht ist.

```
20 d=0:fors=1tolen(z$):h=asc(mid$(z$,s))
-48:d=d*b+h+7*(h>9):next
50 rem in 20 beliebig in dez
```

Die Fakultätsfunktion

Dieser Einzeiler von Detlev Marks berechnet Fakultäten besser als ein Taschenrechner (größer als »69«). Der C 64 besitzt standardmäßig keinen Befehl zur Berechnung der Fakultät.

Die Eingabevariable ist A. B dient als Zählvariable und C als Rechen- und Ausgabe-Variable.

```
10 rem fakultaeten
20 :
30 inputa:frb=1toa:c=c+log(b):next:c=c/l
og(10):print10↑(c-int(c));"e";int(c):run
40 :
```

Dividieren mit beliebig vielen Nachkommastellen

Die Variable Z enthält die Zahl, die geteilt wird, die Variable D den Dividenden, durch den geteilt wird. In N kann die Anzahl der gewünschten Nachkommastellen angegeben werden. Die Variable E% enthält das Ergebnis ohne Rest. Es wird ausgegeben. R enthält den Rest und in Q% steht der Quotient (beschränkt auf Vorkommastellen).

Die Vorkommazahl erhält man durch Q=INT(Z/D), hier vereinfacht durch Q%=Z/D. Q% wird ausgegeben. Dann wird der Rest (Z-Q% * D) berechnet. Die Schleife wird n mal durchlaufen (N=Anzahl der Nachkommastellen).

Analog zu Q% wird E%=R * 10/D berechnet. E% wird ausgegeben als 1. Stelle. Der neue Rest wird gebildet durch: R (neu) = 10 * R (alt) - D * E%.

Nun kann die Schleife durchlaufen werden, bis alle Nachkommastellen ausgegeben wurden.

Beispiel: Wie lautet die 85. Nachkommastelle von 116/13? Z=116, D=13, N=85 sind die Variablenwerte. Das Programm liefert das Ergebnis 9.

Der Autor dieses Programms ist Heinz Bauschke.

```
0 inputz,d,n:q%=z/d:printq%:r=z-d*q%:for
i=1ton:e%=r*10/d:print"▒";e%;:r=10*r-d*e
%:next
10 rem
```

INPUT mit Komma

Diese INPUT-Routine von Jürgen Reinert ersetzt den INPUT-Befehl. Sie erlaubt Komma, Doppelpunkt und Strichpunkt als zusätzliche Satzzeichen bei der Eingabe. Sonst funktioniert sie genauso wie der INPUT-Befehl. Die Routine übernimmt alle Zeichen der Tastatur, auch führende Leerzeichen (Leerzeichen vor Beginn des Textes).

Die Variable AA enthält die aktuelle Eingabe in ASCII-Code. II bildet die Laufvariable für Schleifen und XX$ enthält den eingegebenen Text.

Das »Herz« dieses Einzeilers ist die Eingaberoutine ab Adresse 42336. Diese schreibt alle 80 Zeichen einer Bildschirmzeile in den Basic-Eingabepuffer, der bei Adresse 512 beginnt. Dann liest das Programm Zeichen für Zeichen den Eingabepuffer bis zur genannten 0 und stellt daraus den String XX$ zusammen.

Erfolgt keine Eingabe, das heißt, nur die RETURN-Taste wurde gedrückt, so wird die Routine mit XX$=CHR$(32) verlassen. In allen anderen Fällen enthält XX$ alle sichtbaren, eingegebenen Zeichen (außer Steuerzeichen).

```
1 SYS 42336:XX$="":FOR II=512 TO 600:AA=PE
  EK(II):IF AA THEN XX$=XX$+CHR$(AA):NEXT
10 :
20 REM INPUT
```

Formatierte Ausgabe

Dieser Einzeiler gibt Zahlen beliebiger Länge und unabhängig vom Vorzeichen rechtsbündig aus. Die Tabulatorfunktion von Volker Walter ist sicher beim Aufbau von Tabellen nützlich.

Die Variable A enthält die Zahl, die ausgegeben wird. Die Zahl »22« legt fest, daß die Kommata der ausgegebenen Zahlen in Spalte 24 stehen (2 Stellen werden vom Komma gebraucht). Die Spaltennummer kann im Listing beliebig gesetzt werden (Zahl im Listing = tatsächliche Spaltennummer – 2).

Ist A>=1, so wird der Zehnerlogarithmus von A berechnet und von der Spaltennummer abgezogen, bei der der Dezimalpunkt stehen soll. Da der C 64 nur den Logarithmus zur Basis e berechnen kann, muß das Ergebnis mit der Konstanten .43429448188 multipliziert werden. Dadurch erhält man den Zehnerlogarithmus.
Diese Berechnung erfolgt durch
INT(LOG(B-(B=0))*.43429448188)*(B<=1).
Ist A<1, so wird zur Spaltennummer eine Eins addiert.
Null bildet einen Sonderfall, da die Null noch vor dem Komma stehen muß.
INT(-B)*(B<1).

```
20 b=abs(a):printtab(int(log(b-(b=0))*.4
3429448188)*(b>=1)+int(-b)*(b<1)+22);a
```

Mehr Struktur mit Spaces

Der Basic-Interpreter auf dem C 64 überliest führende Leerzeichen zwischen Zeilennummer und dem Basic-Befehl einfach. So ist ein Einrücken mehrerer Programmzeilen oder das Einfügen von Leerzeilen, wodurch das Programm wesentlich übersichtlicher würde, standardmäßig nicht möglich.

In der Regel behilft man sich in solchen Fällen, indem man an den Zeilenanfang einen Doppelpunkt setzt, auf den dann die gewünschte Anzahl Leerzeilen folgt.

Von Herbert Heise stammt eine wesentlich elegantere Methode:

Der Basic-Interpreter überliest bei der Zeileneingabe Grafikzeichen. Diese Tatsache kann man sich zunutze machen. Nach der Zeilennummer wird irgendein Grafikzeichen (besonders einfach: SHIFT und gleichzeitig die Leertaste) getippt, danach die gewünschte Anzahl führender Leerzeichen und dahinter die vorgesehenen Basic-Befehle. Auf dem Bildschirm ist das Grafikzeichen noch zu sehen. Aber beim Auflisten stellt man fest, daß das Grafikzeichen nicht in den Programmtext übernommen wurde. Dennoch erscheint die Zeile um die gewünschte Anzahl von Stellen eingerückt.

Um ganze Leerzeilen mit dieser Methode zu erzeugen, schreibt man hinter die Zeilennummer ein Grafikzeichen, ein Space und noch ein Grafikzeichen. Nach einem RETURN wird nur das Leerzeichen in die Zeile übernommen. Mit LIST erhält man nun eine Zeile, die nur aus einem Leerzeichen besteht.

In diesem Zusammenhang ist noch folgender POKE-Befehl interessant:

Mit »POKE 129,58« werden Basic-Zeilen nicht mehr ausgeführt, die keine Spaces in Anführungszeichen enthalten. Der Interpreter meldet »?SYNTAX ERROR«. Ausschalten läßt sich dieser Modus wieder mit »POKE 129,322«.

Zeilen löschen am Bildschirm

Diese kleine Routine von Stefan Keimeier löscht bestimmte Zeilen auf dem Bildschirm. Dabei wird eine Maschinenroutine des C 64 verwendet, die diejenige Zeile vom Bildschirm löscht, deren Zeilennummer im X-Register steht. Die Zeilennummer wird zunächst in die Speicherzelle gePOKEt, deren Inhalt der SYS-Befehl in das X-Register übernimmt. Dabei wird von 0 bis 24 gezählt. Die Position des Cursors bleibt davon unbeeinflußt.

Die Variable LN enthält die Zeilennummer (0 bis 24), V gibt die Von-Zeile und B die Bis-Zeile an (gelöscht wird »von« »bis«).

```
10 FOR LN=V TO B:POKE 781,LN:SYS 59903:NEXT
20 :
30 REM FUER NUR EINE ZU LOESCHENDE ZEILEGILT:
40 :
50 POKE 780,LN:SYS 59903
60 :
70 REM                  STEFAN KEIMEIER
```

Grafikbildschirm löschen mit DIM-Befehl

Die Variablen A und B müssen vor der DIM-Anweisung angelegt werden:
(A=0:B=0), da im Programmverlauf das Variablenfeld beeinflußt wird.
Dann werden durch
A=PEEK(49):B=PEEK(50) die Werte für das Variablenende gesichert. Durch das Dimensionieren einer Variablen wird ein entsprechend großer Platz hinter dem Variablenende auf Null gesetzt.

Der Bildschirmspeicher fängt bei Adresse 8192 und geht bis Adresse 16191. Hinter dem bisherigen Variablenende (Adresse: A+B*256) wird die Variable F (auf der Adresse 16191-a-b*256) angelegt. Pro indizierter Variablen werden 5 Byte freigemacht. Dabei wird die mit 0 indizierte Variable nicht berücksichtigt, ebenso die ersten sieben Byte für Variablenname und Dimension.

Durch POKE 49,A:POKE 50,B werden die alten Werte wieder hergestellt. F ist nun nicht mehr dimensioniert.

Wenn das Variablenende größer oder gleich 8186 ist, wenn also das Programm einschließlich Variablenfeld größer ist als 6 Bytes, dann wird der Bildschirm nicht ganz gelöscht.

Soll das Grafikprogramm zum Speichern von Bildern benutzt werden, darf man nicht vergessen, den Zeiger für Speichergrenzen (PEEK(55)+PEEK(56)*256) auf eine Adresse kurz hinter den Bildschirmspeicher zu setzen.

Der Autor des Programms, Manfred Hedtke, schlägt vor, im Grafikprogramm die Zeile 10 immer dann aufzurufen, wenn über die Tastatur »SHIFT + CTRL HOME« (entsprechend »freier Bildschirm im Textmodus«) eingegeben wird.

```
10 a=0:b=0:a=peek(49):b=peek(50):dimf((1
6191-a-b*256)/5):poke49,a:poke50,b
20 rem
```

Soft-Scrolling

Mit diesem Einzeiler von Georg Brandt kann ein beliebiger Text von rechts nach links punktweise über den Bildschirm geschoben werden. Die Variable A mit dem Wert 53270 bildet das Register für horizontales Smooth-Scrolling. Die Variable L enthält die Anzahl der Zeichen (L=40 für die gesamte Bildschirmbreite, weniger für kleinere Textausschnitte), die gleichzeitig auf dem Bildschirm erscheinen sollen. In der Variablen A$ erwartet die Routine den zu zeigenden Text. Das letzte Zeichen sollte ein Leerzeichen sein. Das rechtsbündige Zeichen des Strings steht zweimal auf dem Bildschirm, da es nicht gelöscht wird.

Das Programm arbeitet nach folgendem Prinzip: Der Text wird auf den Bildschirm ausgegeben. Dann wird der Bildschirminhalt mit Hilfe des Smooth-Scrolling-Registers punktweise nach links gezogen, bis er um sieben Punkte verschoben ist. Nun wird der gesamte Text nach links geschoben und das Scroll-Register zurückgesetzt. Dadurch scheint der Text um den achten Punkt verschoben zu sein.

Wichtig:
- Vor Programmaufruf sollte der Bildschirm gelöscht werden, da sonst auch der restliche Bildschirminhalt verschoben wird.
- Auch andere Steuerzeichen außer dem hier verwendeten HOME können eingesetzt werden, um den Text zu positionieren.

```
1 FOR R=1 TO LEN(A$):FOR I=207 TO 200 STEP
  -1:PRINT"{HOME}"MID$(A$,R,L):POKE A,I:NE
  XT I,R
10 :
20 REM SCROLL
```

Scrollen in x-Richtung

Bei diesem Einzeiler von Hans-Peter Harmann kann die Geschwindigkeit des Scrollers je nach Anwendung neu eingestellt werden. Die Verzögerungsschleife

FOR G=0 TO 3:NEXT

muß nur entsprechend abgeändert werden. Durch den Wert 3 wird nur eine minimale Verzögerung erreicht. Experimentieren Sie ruhig mal damit, mit welchen Verzögerungswerten der Bildschirm wie schnell gescrollt wird.

```
10 fort=1to7:poke53270,t:forg=0to3:next:
next:onagoto10:fory=1024to2023:pokey,194
:next:a=1:goto10
20 rem
```

Scrollen bei bleibendem Text

Dieses kleine Programm stammt von Peter Eckart. Es bewirkt, daß nur Teile des Bildschirmes gescrollt werden. Die Zahl, die Sie in Speicherzelle 59639 POKEn, legt fest, wieviele Zeilen am oberen Rand stehen bleiben. So kann man die Kopfzeile einer Tabelle auf dem Bildschirm festhalten oder eine Information, die für ein Programm wichtig ist (Tastenbelegung, Erklärung der wichtigsten Befehle oder anderes). Beendet wird dieses Programm durch den Befehl POKE 1,55.

Wenn Sie eine »0« gePOKEt haben, wird eine Zeile auf dem Bildschirm festgehalten. Beim POKEn einer »1« zwei Zeilen, beim POKEn einer »3« ...

Obwohl diese Routine sehr kurz ist, braucht sie sehr lange zur Ausführung.

```
1 fori=40960to49151:pokei,peek(i):next:f
ori=57344to65535:pokei,peek(i):next:poke
59639,10:poke1,53
```

Grafik: Rahmen zweifarbig

Der Bildrahmen des C 64 ist standardmäßig einfarbig. Von Markus Hillebrand stammt dieser Einzeiler, der einen zweifarbigen Außenrahmen simuliert.

So einfach diese Routine ist: sie verblüfft einen doch!

Der Rahmen erscheint in schwarz (0) und weiß (1). Beim Abtippen der Programmzeile ist es ganz wichtig, die Anzahl der Doppelpunkte und der Leerstellen genau zu treffen. Zwischen den beiden POKEs und den Doppelpunkten stehen je fünf Leerzeichen. Danach folgen 17 beziehungsweise 15 Doppelpunkte. Wichtig ist dies deshalb: Der Autor des Programms wollte die Grenze zwischen den beiden Farben möglichst ruhig auf dem Bildschirm haben. Dazu mußte er die genaue Farbwechselperiode finden. Da der Interpreter Doppelpunkte und Leerzeichen unterschiedlich schnell abarbeitet, hat er die Feineinstellung mit Leerzeichen vorgenommen.

Aber Vorsicht: Sobald während des Programmablaufs eine Taste gedrückt wird, fängt das Bild an zu »laufen«, weil die Raster-Interrupt-Tastaturabfrage in Kraft getreten ist, die den Zeitplan durcheinander bringt.

```
10 poke53280,1     :::::::::::::::::poke
53280,0     :::::::::::::::goto10
20 rem
```

Spiralen mit dem Plotter 1520

Dieser Einzeiler malt mit dem Plotter 1520 eine Spirale, die aus lauter Dreiecken besteht. Zugrunde liegt diesem Programm von Christoph von Rhein die Berechnung eines Kreises. Die X- und Y-Koordinaten werden mit den Funktionen SIN und COS errechnet. Da X und Y mit der Laufvariablen I multipliziert werden, kommt bei dieser Berechnung eine Spirale heraus. Wenn Sie das Aussehen der Grafik ändern wollen, dann können Sie den Faktor 2 innerhalb der Sinus- und Cosinus-Klammern durch einen anderen Wert ersetzen.

```
1 OPEN 1,6,1:PRINT#1,"M",240,0:FOR I=1 TO
  250:PRINT#1,"S",240+SIN(I*2)*I,COS(I*2)*
  I:NEXT
```

Ein grafischer Disassembler: DI-AS

Der Speicherbereich des C 64 läßt sich in 256 Seiten à 256 Byte aufteilen. DI-AS interpretiert die Speicherinhalte als Bild-



schirmcode und stellt die 64 KByte des C 64 Seite für Seite auf den Zeilen 7-13 des Bildschirms dar. Die Seitennummer wird am oberen Bildschirmrand angezeigt. Andreas Carl, der Autor dieses Programms, sagt zur Bedienung des Dissassemblers:

Bevor das Progamm geladen und gestartet wird, muß mit »SYS64738« unbedingt ein Reset durchgeführt werden. Dann sollte der Bildschirm mit »CLR« gelöscht werden. Mit »RUN« wird das Programm gestartet. Dann stehen dem Benutzer die folgenden Optionen offen:
- im Speicher vorwärts blättern (CURSOR-RIGHT)
- im Speicher rückwärts blättern (CURSOR-DOWN)
- die Seite NR betrachten (R/S POKE3,NR:CLR:RUN)

Es stehen umfangreiche Fehlerbehandlungen zur Verfügung.

```
5 print"▒▒1▒b▒q▒(▒HP▄▒ :"peek(3)"▒ ":pok
e41,5:geta$:poke3,peek(3)-(a$="▒")+(a$="
▒"):sys1024:goto5
```

Speicherblockverschiebung

Dieser Einzeiler von Jens Baas dient zur Übertragung von Speicherblöcken. Er ist zum Beispiel nützlich, wenn Sie das Basic oder das Betriebssystem vom ROM ins RAM verlegen wollen. Die Variablen mit »L« bezeichnen jeweils das Low-Byte der Adresse, die Variablen mit »H« das High-Byte.
Für eine Adresse X lassen sich diese so berechnen:
AL=X-256*INT(X/256):AH=INT(X/256).
Die Routine benutzt die Blockverschiebe-Routine des ROMs.

```
1 poke95,al:poke96,ah:poke90,el:poke91,e
h:poke88,nl:poke89,nh:sys41919
11 rem beispiel:
12 poke95,0:poke96,160:poke90,0:poke91,1
92:poke88,0:poke89,192:sys41919
13 rem ↑↑ basic ins ram ↑↑
```

Merge

Bisher gab es oft nur die eine Möglichkeit, wenn ein Programm aus zweien zusammengesetzt werden sollte: das kürzere mußte ans Ende des längeren getippt werden. Hier ist eine ganz kurze Routine von Andreas Gast, die Programme verkettet.

```
10 a=peek(45)+256*peek(46)-2:poke44,a/25
6:poke43,a-peek(44)*256:print"prg laden
& p043,1:p044,8
```

Directory laden, ohne das Basic-Programm zu zerstören

Jedem Programmierer ist das Problem, das diesem Einzeiler zugrunde liegt, sehr bekannt: Man hat - mit Ach und Krach - ein Basic-Programm geschrieben. Nun will man seine Arbeit speichern und erkennt seine freien Disketten nicht wieder. Wenn man die Disketten aber mit
LOAD "$",8 : LIST
listet, löscht man das neue Programm und die ganze Arbeit war umsonst. Reinhard Abdel-Hamid schlägt vor, die folgende Programmzeile vor dem eigenen Programm einzugeben. Mit ihr läßt sich das Directory listen. Das im Speicher liegende eigene Programm bleibt erhalten.

Gestartet wird mit
OPEN 1,8,2,"$":GOTO 0.

Der OPEN-Befehl eröffnet eine sequentielle Datei (hier das Directory) zum Lesen. Mit
GET #1,A$
wird ein Byte vom Disketten-Puffer geholt und das Kommando
A=ASC(A$+"SHIFT/HOME")
wandelt ASCII-Zeichen in Zahlen (von 0 bis 255) um. Hat A$ die Länge 0, so behält der Ausdruck trotzdem die Länge 1.
Durch
PRINT CHR$((A=130 AND 13 OR ((31<A AND A<95) AND A)));
wird das zugehörige Zeichen zu A gedruckt, wenn A einen Wert zwischen 32 und 95 hat. Ist A=130, so wird RETURN ausgegeben. Die Steuerzeichen und Grafiksymbole werden durch die Formel innerhalb der Charakterstring-Klammer »herausgefiltert«. Nur Zahlen, Buchstaben und Satzzeichen werden ausgedruckt.

```
0 get#1,a$:a=asc(a$+"▒"):printchr$((a=13
0and13or((31<aanda<95)anda));:goto0
```

Die Routine kann auch für andere Zwecke angewendet werden.

Es ist ohne weiteres möglich, alle Kommentare, Inhalte von Print-Anweisungen und Texte des auf der Diskette befindlichen Programms auf den Bildschirm zu bringen, da man alle Programme auf der Diskette als sequentielle Datei lesen kann.

Eröffnen Sie die Datei dafür nach dem folgenden Schema:
OPEN 1,8,2,"filename"
Dann wird das Programm mit
GOTO 0 gestartet. Wenn die Ausgabe beendet ist, wird mit der RUN/STOP-Taste die Endlosschleife wieder verlassen.

Maschinenprogramme speichern

Mit dem folgenden Einzeiler von Markus Eicher können Maschinenprogramme sehr einfach gespeichert werden, ohne den Basic-Pointer zu verstellen. Die Werte für LE und HE werden berechnet, indem man auf die Endadresse eine 1 addiert.
Le = Low-Byte der Endadresse
He = High-Byte der Endadresse
La = Low-Byte der Anfangsadressse
Ha = High-Byte der Anfangsadresse
Low- und High-Byte einer Adresse werden so berechnet:
Adresse = dez. 2000
Highbyte = INT(2000/256)=78
Lowbyte = 20000-Lowbyte*256=32

Die Variable A$ enthält den Namen der Datei, in die auf der Diskette das Maschinen-Programm abgelegt werden soll. X legt fest, ob das Programm auf Diskette (X=8) oder Kassette (X=1) gespeichert wird.

Falls ein langer Programmname gewählt wird, müssen die Befehle abgekürzt werden.

```
1 sys(57812)a$,x:poke193,ls:poke194,hs:p
oke174,le:poke175,he:sys62957
```

Beispiel 1:
Das Programm auf den Adressen 20000 bis 22000 soll auf der Diskette unter dem Namen »Beispiel1« gespeichert werden. Der Dateiname wird der Variablen $A zugewiesen.
Die POKE-Befehle lauten

64er-online.de

POKE193,32:POKE194,78;POKE174,241:POKE175,85

Beispiel 2:

Ein Programm von $C000-$C37E soll auf Kassette gespeichert werden. Die Variable X hat also den Wert 1.

Die Low- und High-Bytes sind:

1. von $COOO
   Low = $00 = dez. 0
   High = $C0 = dez. 192
2. von $C37E
   Low = $7E = dez. 126
   High = $C3 = dez. 195

Die POKE-Befehle heißen:

POKE193,0:POKE194,192:POKE174,127:POKE175,195

Floppy-Zeit verkürzen

Die kleine Routine von Robert Loos dient dazu, die Zugriffszeit der Floppy-Disk 1541 ganz entscheidend zu verkürzen. Eine sichere Funktion der Floppy wird nicht gefährdet, wenn der Schrittmotor, der den Schreib-Lesekopf bewegt, wesentlich schneller arbeitet. Der Schrittmotor wird im Interrupt bedient. Daher genügt es, die Größe des Interruptintervalls zu verändern, um die Drehzahl des Motors zu beeinflussen. Standardmäßig wird etwa alle 15 Millisekunden (hier etwa alle 4 Millisekunden) ein Interrupt ausgelöst, der den Stepper um eine Viertelspur bewegt. Alle Bewegungen des Kopfes werden dadurch fast viermal schneller. Vorteile sind neben der Zeitersparnis:

Das Laufgeräusch wird angenehm leise und kurz, und im Falle einer Kopfjustage fährt der Kopf mit erheblich verminderter Kraft gegen den Anschlag, sodaß die Gefahr einer Dejustage deutlich gemindert ist.

```
10 OPEN 1,8,15,"M-W"+CHR$(7)+CHR$(28)+CHR$
   (1)+CHR$(15)
20 :
30 REM ZUGRIFFSZEIT DER FLOPPY KUERZER
```

Ein RENEW, das funktioniert!

Noch ein Programm von Hartmut und Martin Sprave: Dieser Einzeiler ist im Direktmodus, also ohne Zeilennummern, einzugeben. Vorher dürfen keine Variablen definiert oder Basic-Zeilen eingetippt werden, da sonst das gelöschte, aber noch im Speicher befindliche Programm zerstört wird. Im Wesentlichen besteht das RENEW-Programm auf dem Aufruf einer System-Routine, die die Basic-Zeilen neu bindet und das Ende des gelöschten Programms herausfindet. Das erste POKE tut so, als ob sich ein (ungelöschtes) Basic-Programm im Speicher befindet, weil sonst die Routine nicht arbeitet. Die Endadresse des Basic-Programms wird um zwei erhöht und in den Zeiger auf den Start der Variablen (45/46) übertragen. Der CTR-Befehl gleicht alle weiteren Basic-Zeiger diesem Befehl an.

```
6 rem c-64
7 poke2050,8:sys42291:poke46,peek(35)-(p
eek(781)>253):poke45,peek(781)+2and255:c
lr
```

Ein einfaches Renumber

Diese Routine von Georg Wichert numeriert ein Basic-Programm, das bis zu 255 Zeilen hat, in wenigen Sekunden neu. Die erste Zeilennummer ist dann 0, die Schrittweite ist 1, Sprungadressen der Befehle GOTO und GOSUB bleiben beim Neunumerieren unverändert. Jedes Basic-Programm belegt die Speicherplätze 2048 bis PEEK(45)+PEEK(46)*256-3. PEEK(2049)+PEEK(2050)*256 gibt an, bei welcher Adresse die erste Zeile aufhört. Die erste Zeilennummer A wird aufgrund der Adressen 2051 und 2052 folgendermaßen berechnet:

A=PEEK(2051)+PEEK(2052)*256.

Die zweite Zeilennummer findet man im Speicher an den Adressen

PEEK(2049+PEEK(2050)*256+2 und

PEEK(2049)+PEEK(2050)*256+3.

Die Zeilennummern befinden sich jeweils an den Adressen A+2 (Low Byte) und A+3 (High Byte).

Ist das Basic-Programm, das neu numeriert wird, länger als 255 Zeilen, dann muß der Einzeiler zum Zweizeiler abgeändert werden. Dann wird folgendes wichtig: Aus Platzgründen wird in dieser Programmversion an die Stelle A+2 der Wert z gePOKEt (z=0,1,2...,n-1; n-1 steht für die Anzahl der Zeilen des Basic-Programms, das umnumeriert werden soll, inklusive dem Einzeiler) und an die Stelle A+3 den Wert 0.

```
1 FOR A=2049 TO PEEK(45)+PEEK(46)*256-3:PO
  KE A+2,Z:POKE A+3,0:A=PEEK(A)+PEEK(A+1)*
  256-1:Z=Z+1:NEXT
10 :
20 REM RENUMBER (GEORG WICHERT)
```

Track-Zerstörer: Kopierschutz

Diese Programmzeile von Jörg Wegmeyer produziert auf dem gewünschten Track der Diskette (Variable T) einen READ ERROR 21. Damit erreicht man einen relativ sicheren Kopierschutz.

```
1 open1,8,15:open2,8,2,"#":print#1,"u1 2
 0";t;0:print#1,"m-e"chr$(163)chr$(253)
```

Eine Zeile = kompletter Datenschutz

Diese Routine von Volker Ritzhaupt erreicht, daß auf einer Diskette weder das Inhaltsverzeichnis gelistet, Programme gelöscht noch irgendetwas auf die Diskette geschrieben werden kann. Das Laden der Programme funktioniert hingegen ganz normal. Der Inhalt der Diskette kann nur durch Formatieren mit einer neuen ID-Nummer noch manipuliert werden.

Das Directory wird nicht mehr gelistet, weil es als Basic-Programm geladen wird und in dem veränderten Directory drei Nullen am Anfang (für den Interpreter das Zeichen für Programmende) erscheinen. Das »Directory-Programm« endet also schon nach zehn Bytes. Die ersten fünf Byte stellen den Zeilenanfang und die Zeilennummer dar. Darauf folgen, wie immer am Anfang eines Inhaltsverzeichnisses, ein Leerzeichen und ein Anführungszeichen. Um diese drei Zeichen beim Listen verschwinden zu lassen, folgen nun drei chr$(20), also drei DELETEs. Dies bewirkt, daß das Directory beim Listen völlig verschwindet. Da nun unmittelbar die drei Nullen folgen, wird das Listen hier abgebrochen. Auf den LIST-Befehl erscheint nur die Meldung »READY«.

```
1 open1,8,3,"#":open2,8,15,"b-p3,144":pr
int#1,"[illegible]"chr$(0)chr$(0)chr$(0):print#2
,"u2:3,0,18":print#2,"i
2 rem
3 rem verhindert auflisten des directory
      und macht schreibschutz auf disk
4 rem
```


64er-online.de

Der Befehl GOTO X – eine Basic-Erweiterung

Dieses Programm von Reinhard Jurk simuliert den im Commodore-Basic nicht vorhandenen Befehl GOTO X.

Zu dem etwas seltsamen Aussehen der Programmzeile: In den Ausführungszeichen steht ein Maschinenprogramm, dessen Opcodes im Listing des Basic-Interpreters diese merkwürdigen Zeichen erzeugen.

```
10 syspeek(61)+256*peek(62)+26:"[Steuerzeichen]"
20 rem
30 rem goto x: ll%=zeilennummer:goto10
40 rem
```

```
  BITTE UNBEDINGT GENAUSO EINGEBEN !!!
  43 KLAMMERAFFEN (@)
5 GOTO20
10 SYSPEEK(61)+256*PEEK(62)+26:"@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"
20 FORA=2082TO2124:READX:S=S+X:POKEA,X:N
EXT:IFS<>5185THENPRINT"DATA ERROR !":END
30 PRINT"OK."
40 DATA169,204,133,69,133,70,165,157,133
,85,133,86,160,12,200,145,85,162,129
50 DATA202,200,138,145,85,172,128,163,16
6,46,165,45,32,17,177,32,43,175,32,1
60 DATA184,76,163,168
READY.
```

Der Einzeiler als DATA-Lader

```
C*
    PC  IRQ  NV-BDIZC AC XR YR SP
.;  E147 EA31 00110001 08 08 02 F6

.,  081A  A9 CC      LDA #$CC
.,  081C  85 45      STA $45
.,  081E  85 46      STA $46
.,  0820  A5 9D      LDA $9D
.,  0822  85 55      STA $55
.,  0824  85 56      STA $56
.,  0826  A0 0C      LDY #$0C
.,  0828  C8         INY
.,  0829  91 55      STA ($55),Y
.,  082B  A2 81      LDX #$81
.,  082D  CA         DEX
.,  082E  C8         INY
.,  082F  8A         TXA
.,  0830  91 55      STA ($55),Y
.,  0832  AC 80 A3   LDY $A380
.,  0835  A6 2E      LDX $2E
.,  0837  A5 2D      LDA $2D
.,  0839  20 11 B1   JSR $B111
.,  083C  20 2B AF   JSR $AF2B
.,  083F  20 01 B8   JSR $B801
.,  0842  4C A3 A8   JMP $A8A3
```

Die Bedeutung des DATAs im Assembler-Code

Zur Eingabe:
In diesem Listing ist ein Generierungsprogramm angegeben. Dieses wird eingegeben und gestartet. Wenn der Checksummer-Test positiv ausgefallen ist und man sich vergewissert hat, daß das Programm richtig eingegeben wurde, gibt man »LIST« ein, gefolgt von »NEW«. Dann fährt man mit dem Cursor auf Zeile 10 und drückt RETURN. Die Routine steht nun im Speicher.

Zum Einbinden der Routine in eigene Programme ändert man die Zeilennummern entsprechend ab, weist der Variablen LL% den Wert der anzuspringenden Zeile zu und springt mit »GOTO« zu der Zeile, in der GOTO X steht.

File-Einträge mit Sonderzeichen

Mit dem Einzeiler von Ralf Peiler lassen sich Zusätze an Filenamen im Directory anhängen. So wird zum Beispiel aus dem Directory-Eintrag "name" der Directory-Eintrag "name",8: oder "name",8,1. Doch nun der Einzeiler:

```
INPUT"name";A$:OPEN 1,8,15, "R:"+A$+"{SHIFT-SPACE}
{COMMODODRE-D}8{SHIFT-KLAMMERAFFE}="+A$:CLOSE 1
```

»SHIFT-SPACE« teilt dem DOS mit, ein Anführungszeichen zu setzen.

»COMMODORE-D« ist der Code für »,«. »8« entspricht der normalen »8« und »SHIFT-KLAMMERAFFE« ist der Code für »:«. Soll »,8,1« an den Filenamen angehängt werden, so ist »{COMMODORE-D}8{SHIFT-KLAMMERAFFE}« zu ersetzen durch »{COMMODORE-D}8{COMMODORE-D}«. Wie Sie sehen, geht das Anhängen kinderleicht. Verblüffung kann man auch erzeugen durch ein buntes Directory. Allerdings lassen sich nicht sämtliche Farben erzeugen. Bisher sind folgende Kombinationen gefunden worden:

```
name {SHIFT-SPACE} {CTRL-2} = name → weiß
name {SHIFT-SPACE} {CTRL-3} = name → rot
name {SHIFT-SPACE} {CTRL-4} = name → blau
name {SHIFT-SPACE} {CTRL-6} = name → grün
name {SHIFT-SPACE} {CTRL}   = name → clr
name {SHIFT-SPACE} {DEL}    = name → del (auch mehrmals)
name {SHIFT-SPACE} {RVSON}  = name → bis »PRG« revers
```

Trick 17 mit ON ... GOTO

Eine sehr interessante Version einer ON...GOTO-Anweisung stammt von Peter Zankl.

Der Ausdruck (A$="A") hat den Wert -1, wenn ein A eingegeben wurde. Sonst hat er den Wert 0. Rechnen Sie nach oder probieren Sie es aus! (cg)

```
100 rem tastaturabfrage mit sprung
200 rem :
300 rem vorher:
400 :
410 geta$:ifa$=""then410
420 ifa$="a"then2000:rem programmteil a
430 ifa$="b"then3000:rem programmteil b
440 ifa$="x"then end:rem ende
450 goto 410
499 :
500 rem nachher:
600 :
610 geta$:on1-(a$="a")-2*(a$="b")-3*(a$=
"x")goto610,2000,3000:end
620 :
```

20000 Byte mehr

20 KByte adressierbarer Speicher werden uns beim normalen Basic vom Computer vorenthalten. Dieser Speicherplatz ist jedoch nicht tot, er kann mit ein paar Tricks für alle möglichen Zwecke nutzbar gemacht werden.

Der C 64 besitzt zwei Arten von Speicher-Chips. Zum einen den RAM-Speicher (Random Access Memory), zum anderen den ROM-Speicher (Read Only Memory). Auf das RAM kann beliebig zugegriffen werden. Das heißt, man kann diesen Speicher beliebig beschreiben und lesen. Dieser Speicher dient zur Aufnahme der eigenen Programme. ROM heißt soviel wie Nur-Lese-Speicher. Er enthält ein festes Programm oder Daten (Zeichensatz etc.) und kann nur gelesen werden. Der ROM-Speicher behält seine Informationen auch nach dem Ausschalten des Computers. Beim Einschalten des Computers oder nach einem Reset verzweigt der Prozessor ins ROM (Betriebssystem) und initialisiert das gesamte System.

Der C 64 besitzt 64 KByte RAM und 20 KByte ROM. Im ROM sind Basic, Betriebssystem und Standardzeichensatz gespeichert (Bild 1). Für die Ein-/Ausgabevorrichtungen benötigt der C 64 nochmal 4 KByte Speicherkapazität. Ein 8-Bit-Computer mit 16-Bit-Adreß-Bus kann aber normalerweise nur 64 KByte (2 hoch 16 Bit) adressieren. Wo liegt aber der Unterschied zwischen C 64 und den anderen 8-Bit-Computern der 65xx-Reihe?

Das Geheimnis liegt im 6510-Prozessor. Der Chip verfügt im Gegensatz zum 6502 über ein Prozessorregister. Über diesen Port wird gesteuert, ob RAM, ROM oder Ein-/Ausgabe in bestimmten Speicherbereichen aktiv sind. Hierfür benötigt das Register (Speicherstelle 1) allerdings nur 3 Bit. Weitere 3 Bit steuern den Kassettenport. Die restlichen 2 Bit sind unbesetzt und werden einfach überlesen. Das Datenrichtungsregister für diesen Port liegt im Speicherplatz 0.

Die einzelnen Bits im Kontrollregister des 6510 sind wie folgt definiert:

Name	BIT	Funktion
LORAM	0	Basic-ROM/RAM ($A000 bis $BFFF)
HIRAM	1	Betriebssystem-RAM ($E000 bis $FFFF)
CHAREN	2	I/O-Zeichen-ROM ($D000 bis $DFFF)
	3	Schreibleitung
	4	Schalter
	5	Motorsteuerung (Bit 3 bis 5 betreffen nur Kassettenport)
	6	(unbesetzt)
	7	(unbesetzt)

Zu jedem Daten-Port gehört auch ein Datenrichtungsregister. Dieses liegt für den Prozessor-Port in Speicherstelle 1. Der Normalwert für das Datenrichtungsregister lautet 47 oder Binär 00101111, wobei 0 für Eingabe und 1 für Ausgabe steht. In Standardmodus stehen LORAM, HIRAM, CHAREN, Schreibleitung und Motorsteuerung auf »Ausgabe«. Nur der Kassettenschalter steht auf »Eingabe«.

Die Steuerbits des Prozessorregisters erfüllen im allgemeinen folgende Funktion:

BIT 0 (LORAM):
Dieses Bit steuert die Belegung des Speicherbereichs von $A000 Bis $BFFF. Ist Bit 0 gesetzt, so belegt der 8-KByte-Basic-ROM den Adreßbereich. Bringt man diese Bit auf Low, so verschwindet das Basic-ROM und wird durch 8 KByte RAM-Speicher ersetzt.

BIT 1 (HIRAM):
Ist dieses Bit gesetzt, hat der Prozessor im Adreßbereich $E000 bis $FFFF Zugriff auf das Betriebssystem (Kernel-ROM) des Computers. Bei Low-Pegel wird das Kernel durch ebenfalls 8 KByte ersetzt.

Bit 2 (CHAREN):
Dieses Bit ist dafür verantwortlich, ob sich das 4K-Byte-Zeichengenerator-ROM oder das I/O-RAM im Prozessor-Adreßbereich von $D000 bis $DFFF befindet. Wenn Bit 2 gesetzt ist, dann hat der Prozessor Zugriff auf folgende Ein-/Ausgabe-Register:

$D000-$D3FF	:	VIC
$D400-$D7FF	:	SID
$D800-$DBFF	:	Farb-RAM
$DC00-$DCFF	:	CIA1
$DD00-$DDFF	:	CIA2
$DE00-$DEFF	:	freier I/O-Port #1
$DF00-$DFFF	:	freier I/O-Port #2

Ist Bit 2 gelöscht, kann der Prozessor aus dem Zeichen-ROM lesen. In diesem Fall sind die Ein-/Ausgabe-Register gesperrt. Im Normalzustand ist dieses Bit gesetzt. Dieses Bit muß nur gelöscht werden, wenn man den Zeichensatz ins RAM kopieren will. Dabei darf aber das Programm nicht unterbrochen werden, da ja die CIA1 (Tastatur) abgeschaltet ist. Um Bit 2 von Basic aus auf Low zu setzen, muß vorher der Interrupt gesperrt werden, da die Interruptroutine statt des Timers im CIA den Charaktergenerator vorfindet und dadurch abstürzt.

Das Zeichen-ROM kann wie folgt eingeschaltet werden:

```
10 POKE 56334,PEEK(56334) AND 254: REM INTERRUPT
   AUS
20 POKE 1,PEEK(1) AND 251: REM ZEICHEN-ROM EIN
30 Programm.....(Kopie des Zeichensatzes etc.)
1000 POKE 5634,PEEK(56334) OR 1: REM INTERRUPT EIN
```

Ausgeschaltet wird die Sache mit folgendem Programm:

```
10 POKE 56334,PEEK(56334) AND 254: REM INTERRUPT
   AUS
20 POKE 1,PEEK(1) OR 4 : REM I/O-RAM EIN
30 POKE 5634,PEEK(56334) OR 1: REM INTERRUPT EIN
```

Sie werden sich nun aber fragen, wie der Computer trotz Tastaturabfrage Zeichen am Bildschirm ausgeben kann. Dies geschieht mit Hilfe des sogenannten ROM-IMAGE. Weil der VIC-II-Chip gleichzeitig nur auf 16 KByte zugreifen kann, muß der Zeichensatz immer in der jeweiligen Bank des VIC-II-Chips liegen. Im Standardmodus greift der VIC auf Bank 0 zu ($0000 bis $3FFF). Das System ist so entwickelt, daß VIC-II davon ausgeht, daß sich der Zeichensatz im »zugriffsberechtigten« Speicherbereich befindet. Arbeitet der VIC-II in Bank 0, nimmt er an, der Zeichensatz befindet sich ab $1000, obwohl dieser in Wirklichkeit in $D000 liegt. Der Speicherbereich $1000 bis $1FFF ist aber trotzdem für den Anwender als RAM-Bereich vorhanden. Gleiches gilt für die drei anderen möglichen Banks des VIC-II:

Bank	VIC-II-Chip	ROM-IMAGE
0	$0000-$3FFF	$1000-$1FFF
1	$4000-$7FFF	$5000-$5FFF
2	$8000-$BFFF	$9000-$9FFF
3	$C000-$FFFF	$D000-$DFFF

Wenn man nun Bit 1 und 2 löscht, müßte man theoretisch auf 64 KByte RAM Zugriff haben. Das probieren wir gleich mal aus (vorher eventuelle Programme sichern!):

```
POKE 1,PEEK(1) AND 252
```

Jetzt stehen die 64 KByte RAM voll zur Verfügung, nur ist der Computer jetzt »scheintot«. Er reagiert weder auf Tastendruck noch auf gut zureden. Hier hilft nur noch ein Reset oder Ausschalten.

Was passiert bei PEEK und POKE?

Befindet sich der C 64 im Standardmodus (wie nach dem Einschalten), dann sind Basic-ROM und Kernel-ROM aktiv. Wie wir wissen, kann man den ROM-Speicher nicht beschreiben (Read Only Memory). Was passiert nun, wenn man versucht, in diesen Speicherbereich einen Wert zu POKEn? Der Befehl wird nicht einfach ignoriert, wie man das so vom VC 20 oder den anderen CBM-Computern kennt. Der Prozessor schiebt den Wert einfach in das darunterliegende, »versteckte« RAM. Versucht man mit PEEK (X) das Byte wieder zu lesen, stellt man fest, daß der Prozessor auf das ROM zugreift.

Auf diese Weise läßt sich einfach das Basic- und Kernel-ROM ins RAM kopieren. Setzt man anschließend Bit 1 des Prozessorports auf Low, verändert sich scheinbar nichts. Der Prozessor greift jetzt auf das RAM zu und kann beliebig verändert werden.

Im folgenden werden zwei Programme gezeigt, um das Betriebssystem ins RAM zu kopieren und umzuschalten:

```
10 FOR I=40960 TO 49151: POKE I,PEEK(I) : NEXT
20 FOR I=57344 TO 65535: POKE I,PEEK(I) : NEXT
30 POKE 1,53 : REM FÜR C 64
30 POKE 1,5 : REM FÜR SX 64
```

oder wie in Listing 1:

```
>, C000 A9 C0     LDA #$C0
>, C002 85 FB     STA $FB
>, C004 A9 A0     LDA #$A0
>, C006 85 FD     STA $FD
>, C008 20 1E C0  JSR $C01E
>, C00B A9 00     LDA #$00
>, C00D 85 FB     STA $FB
>, C00F A9 E0     LDA #$E0
>, C011 85 FD     STA $FD
>, C013 20 1E C0  JSR $C01E
>, C016 A9 35     LDA #$35
>, C018 85 01     STA $01
>, C01A 8D D6 FD  STA $FDD6
>, C01D 60        RTS
>, C01E A0 00     LDY #$00
>, C020 84 FC     STY $FC
>, C022 B1 FC     LDA ($FC),Y
>, C024 91 FC     STA ($FC),Y
>, C026 C8        INY
>, C027 C0 00     CPY #$00
>, C029 D0 F7     BNE $C022
>, C02B E6 FD     INC $FD
>, C02D A5 FD     LDA $FD
>, C02F C5 FB     CMP $FB
>, C031 D0 EB     BNE $C01E
>, C033 60        RTS
```

Listing 1. So kopiert man das ROM ins RAM

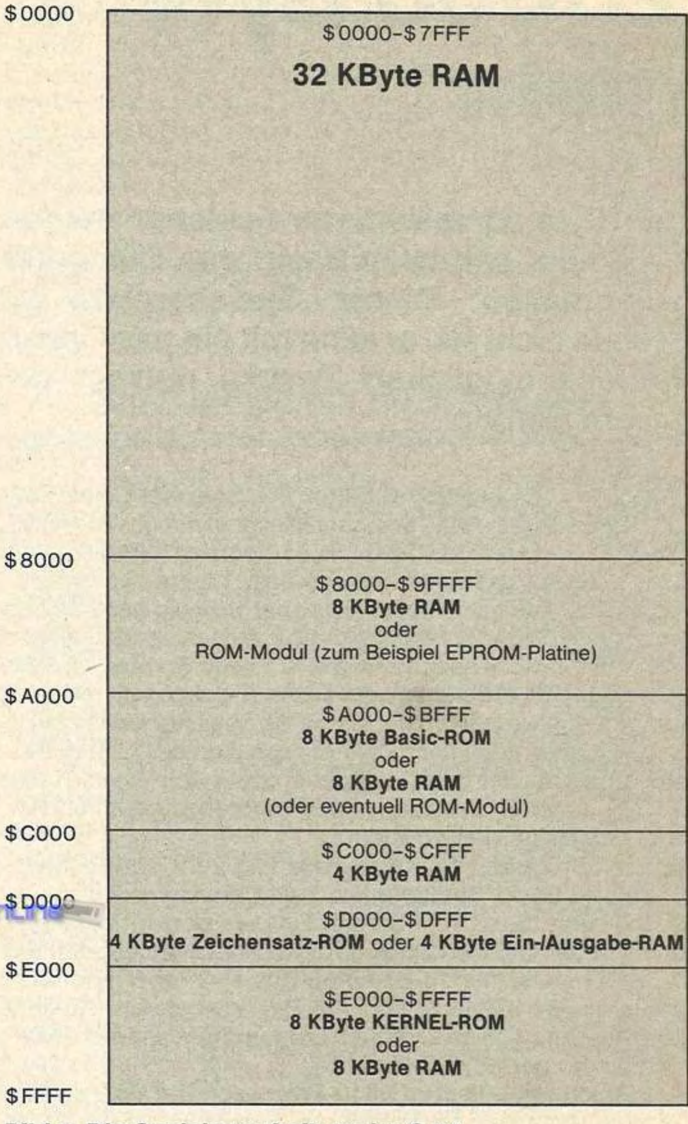


Bild 1. Die Speicheraufteilung im C 64

Hier stellen sich dem Anwender ungeahnte Möglichkeiten, auf die hier nicht im einzelnen eingegangen werden soll.

20 KByte frei – was tun damit?

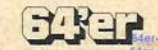
Es gibt auch noch andere Möglichkeiten, den RAM-Bereich unter dem ROM zu nutzen.

Es ist durchaus möglich, zum Beispiel ein HiRes-Grafikbild in diesem RAM-Bereich zu erstellen. Durch Umschalten des VIC-II-Chips steht dem Anwender so sofort ein neues Grafikbild zur Verfügung, während das frühere Bild wieder editiert werden kann. Eine praktische Anwendung hierfür wäre zum Beispiel eine Dia-Show.

Die RAM-Bank unter dem ROM kann auch als sogenanntes RAM-Floppy verwendet werden. Es könnte also durchaus ein 20 KByte langes Maschinen-Programm ohne Schwierigkeiten im »versteckten« RAM untergebracht werden.

Es gibt noch sehr viele Möglichkeiten, den »versteckten« Speicher zu nutzen. Probieren Sie es doch selbst mal. Sie werden erstaunt sein, wo und was ihr C 64 alles speichern kann.

(C.Q. Spitzner/og)



Ping-Pong

Nach all den vielen Tips und Tricks noch ein zusätzliches Bonbon für Sie: Das erste Telespiel der Welt, gespielt auf Ihrem C 64.

Tapfer haben Sie sich durch dieses Sonderheft durchgearbeitet. Zur Entspannung bieten wir Ihnen noch ein kleines Spielchen an. Es ist eine Nachahmung des ersten Telespieles überhaupt: eine einfache Tennis-Simulation. Dabei übt gerade seine Einfachheit einen besonderen Reiz aus. Gesteuert wird das Spiel mit zwei Joysticks, für jeden Spieler einen. Will man alleine spielen, kann man eine Mauer aus reversen Leerzeichen bauen, von der der Ball abgelenkt wird. Man muß hierbei nur darauf achten, daß bis zur Spalte 29 der Ball nur nach rechts, ab der 31. Spalte der Ball nach links abgelenkt wird.

»PONG 64« war ein Beitrag zum Wettbewerb »Bildschirmseite«. Deshalb erscheint das Programm im Listing extrem kompakt und undurchschaubar. Dennoch lassen sich ein paar Parameter verändern. Zum einen ist das die Geschwindigkeit des Balles. Mit POKE 49335,V wird die Geschwindigkeit verändert. Dabei ist 18 der Standardwert. Zum anderen der Winkel des Balles. Ihn flacht man mit POKE 49263,2 ab (normaler Inhalt ist 1). Diese Änderungen sind am besten zwischen den Zeilen 2 bis 8 anzubringen.

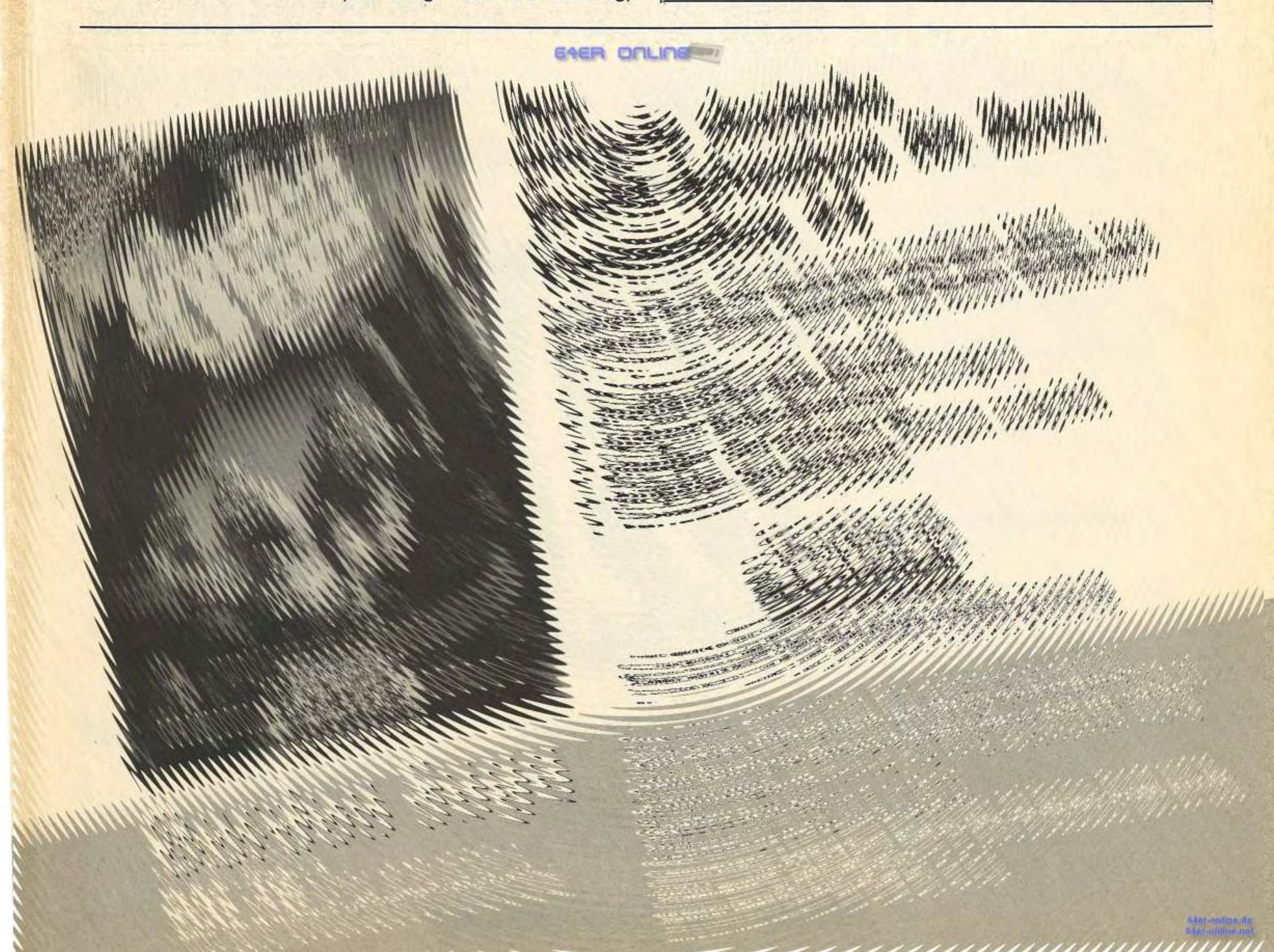
(Tek-Seng The/Ewald Mack/og)

```
0 PRINT"{CLR,RVSON,YELLOW}";:FOR I=0 TO 39
  :PRINT" ";:NEXT:V=49152:FOR J=1 TO 7:REA
  D A$:FOR I=1 TO 36                          <229>
1 POKE V,10*(ASC(MID$(A$,I*2-1,1))-65)+VAL
  (MID$(A$,I*2,1)):V=V+1:NEXT:NEXT            <164>
2 V=832:FOR I=V TO 960:POKE I,0:NEXT:FOR I
  =0 TO 20:POKE V+I*3,255:NEXT:V=53248:POK
  E 2,0                                       <156>
8 POKE V+16,2:POKE V+4,0:POKE V+21,7:Q=(PE
  EK(2)AND 3):P(Q)=P(Q)+1:PRINT"{HOME,RVSO
  N}"P(2);                                    <193>
9 PRINT SPC(25);P(1):SYS 49152:GOTO 8         <243>
50 DATA Q90301A1U801A3U8Q9A001D3U801C3U801
   C9U8Q9H201A2U8Q9D201A0U8Q2B302Y9A702Y8     <114>
51 DATA A7X202Z0A7X202N1A302N4A3Q2A402E0U8
   X202D9U8X202M8A302N7A3X202D2U8Q9R301A2     <090>
52 DATA T201A5T2Q9F001S5T2R4A1U8R3A0W0D2W1
   T202A1U8R4A3U8R3A1W0D2W1T202A3U8U6Z5U7     <252>
53 DATA U8D4Q9A101Z5U7R4A5U8R3Z0T2U8B0U2W4
   F9R6B3X8Z0T2U8A8X2W4Y704A3U6Z0T202A5U8     <062>
54 DATA R3B6U8E1A4R2Y9T2U8B5X8A4U8U8C3H4U8
   G6Q9A601B6U8U8B3U6A4U8U8A8B0Y0F4Q9A201     <227>
55 DATA B6U8Q2B8Q0Z5N6U8Z3U2U8Y8R3D1U8U8A8
   R3D0U8Y0B6U6S5T2R3B6U8X8Y9T2E1A4U8A301     <073>
56 DATA Y9T2H6I1T2K604B1K604B5J6N3A2J6Q9A1
   U8Y9U2W4F8R6Y3X2J6X2W4X004X6U2J6A1A1Z5     <122>
```

© 64'er

»Pong 64«, eine Nachahmung des ersten Telespiels; bitte mit dem Checksummer eingeben

64ER ONLINE



Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger
Leitender Redakteur: Albert Absmeier
Koordination: Georg Klinge
Redaktion: Volker Everts, Christine Geißler, Achim Hübner, Harald Meyer, Markus Ohnesorg, Thomas Röder, Boris Schneider, Arnd Wängler

Titelfoto: Jens Jancke

Layout:
Leo Eder (Ltg.), Sigrid Kowalewski (Cheflayouterin)

Herstellung: Klaus Buck

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-41 56 56, Telex: 862 329
USA: M&T Publishing Inc.; 2464 Embarcadero Way, Palo Alto, CA 94 303

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Leitung Marketing Vertrieb: Hans Hörl (114)
Vertriebsleitung: Helmut Grünfeldt (189)

Anzeigenverwaltung und Disposition: Michaela Hörl

Verlagsleiter M&T-Buchverlag: Günther Frank

Druck: Druckhaus München, Schellingstr. 39-43, 8000 München 40

Preis: Das Einzelheft kostet DM 14,–

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1, Telefon (07 11) 6 48 30

Urheberrecht: Alle in diesem Heft erschienenen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen, vorbehalten. Reproduktionen gleich welcher Art, ob Fotokopie, Mikrofilm oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung des Verlages. Anfragen sind an Michael Scharfenberger zu richten. Für Schaltungen, Bauanleitungen und Programme, die als Beispiele veröffentlicht werden, können wir weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Anfragen für Sonderdrucke sind an Peter Wagstyl (185) zu richten.

© 1986 Markt & Technik Verlag Aktiengesellschaft

Verantwortlich:
Für redaktionellen Teil: Michael Scharfenberger
Für Anzeigen: Brigitte Fiebig
Redaktions-Direktor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

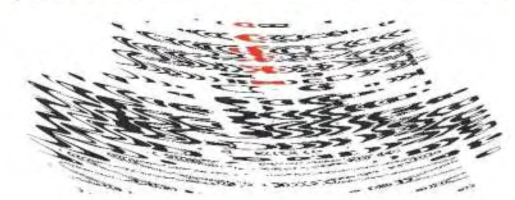
Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 46 13-0, Telex 5-22 052



PROTEXT

64er online

TOP ASS



64er-online.de
64er-online.net



64er-online.de
64er-online.net